REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		FINAL 01 MAR 95 TO 29 FEB 96

4. TITLE AND SUBTITLE
COMPUTATIONAL TECHNIQUES FOR ROBUST AND FIXED-STRUCTURE CONTROL DESIGN

5. FUNDING NUMBERS
F49620-95-1-0244
2300HS 61102F

6. AUTHOR(S)
EMMANUEL G. COLLINS

AFOSR-TR-96
0316

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEPARTMENT OF MECHANICAL ENGINEERING
DFLORIDA A&M UNIVERSITY
2525 POTTSDAMER STREET
TALLAHASSEE, FL 32310-6046

F49620-95-1-0244

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOR/NM
110 DUNCAN AVE, SUITE B115
BOLLING AFB DC 20332-8080

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
F49620-95-1-0244

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITIED.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

SEE REPORT FOR ABSTRACT

19960626 015

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

Final Technical Report
May 30, 1996

Period Covered: 3/1/95 - 2/29/96

Computational Techniques for Robust and Fixed-Structure Control Design

Grant F49620-95-1-0244

Principal Investigator

Emmanuel G. Collins
Department of Mechanical Engineering
Florida A&M University
2525 Pottsdamer St.
Tallahassee, FL 32310-6046
Voice: 904-487-6373
FAX: 904-487-6337
Email: ecollins@eng.fsu.edu

Submitted To:

Air Force Office of Scientific Research
AFOSR/NM
110 Duncan Avenue Suite B115
Bolling Air Force Base
Washington, DC 20332-0001

Table of Contents

1. Introduction

This report describes accomplishments made in four areas of research related to robust, fixed-structure control system design and analysis:

1) the formulation of robust, fixed-structure control design problems in terms of a Riccati equation feasibility and the development of probability-one homotopy algorithms for its solution ;

2) a comparison of descent and continuation techniques for H_2 optimal reduced-order control design and an investigation of the best bases in which to represent the reduced-order controller;

3) the development of parallel processing techniques to implement probability-one homotopy algorithms for reduced-order H_2/H_∞ control design; and

4) the development and implementation of an object-oriented programming approach for the implementation of interior point methods to solve linear matrix inequalities (LMI's).

Below, we motivate each of the four areas of research and briefly describe key accomplishments. The journal paper titles are used as headings and we list both the journal papers that have been submitted and the corresponding conference papers. Each of the journal papers is included in an Appendix.

2. Probability-One Homotopy Algorithms for Robust Controller Synthesis with Fixed-Structure Multipliers

Journal Paper

E. G. Collins, Jr., W. M. Haddad, L. T. Watson, and D. Sadhukhan, "Probability-One Homotopy Algorithms for Robust Controller Synthesis with Fixed-Structure Multipliers," *International Journal of Robust and Nonlinear Control*, accepted for publication.

Conference Paper

E. G. Collins, Jr., W. M. Haddad, and L. T. Watson, "Fixed-Architecture, Robust Control Design Using Fixed-Structure Multipliers," *Proceedings of the International Federation of Automatic Control*, to appear in June 1996.

During the past two decades, major advancements have been made in robust control theory. Building upon H_∞ theory, the structured singular value (SSV) was defined as a nonconservative robustness measure for the analysis of linear systems with arbitrary phase, multiple-block uncertainty. The supremum of the structured singular value over

nonnegative frequencies is the inverse of the multivariable stability margin. The initial developments in structured singular value theory focussed on uncertainty with arbritrary phase (often called "complex uncertainty") and hence, although less conservative than H_∞ theory, could still yield very conservative robustness bounds for systems with parametric uncertainty. This led to the development of mixed (i.e., real and complex) structured singular value (MSSV) theory which considers block-diagonal uncertainty with both complex and real scalar parametric elements.

Parallel research addressed the issue of real parameter uncertainty using absolute stability theory such as Popov analysis and was developed by recognizing the relationship between sector bounded nonlinearities and interval bounds on linear uncertainties. This work was soon seen to provide an upper bound for the MSSV. In fact, in contrast to the initial work on the MSSV, this research provided the first *fixed-structure* multiplier versions of MSSV theory. A unique contribution of some of this work is that it led to the development of upper bounds on an H_2 cost functional over the uncertainty set under consideration. By optimizing this upper bound and using a Riccati equation constraint, continuation algorithms have been developed for MSSV controller synthesis. Note that the H_2 approach allows the direct design of fixed-architecture (e.g., reduced-order or decentralized) controllers and the simultaneous optimization of the controller and (fixed-structure) multipliers, hence avoiding *M-K* (i.e, multiplier-controller) iteration schemes. However, to date the synthesis algorithms have been formulated only for the case of the the Popov multiplier. In addition, the algorithms rely on an initialization scheme, have not used the prediction capabilities obtained by computing the Jacobian of the homotopy (or continuation) map, and have assumed that the homotopy curve is monotonic.

A similar line of research has been developed independently by Safonov, et. al. This work also provides a fixed-structure multiplier version of the MSSV but, unlike the approach described above, this approach develops multipliers for strictly linear uncertainties. The associated robustness analysis was originally formulated in terms of a convex, frequency-domain optimization problem but has recently been reformulated in terms of a (convex) linear-matrix-inequality (LMI) problem. These results have led to the recognition that robust control design can be approached via solving a (nonconvex) "bilinear matrix inequality" (BMI). This approach allows the design of fixed-architecture controllers and can be implemented without using *M-K* iteration. To obtain a reasonably sized BMI, the multiplier set must be restricted to lie in the span of a stable basis. However, the choice of this basis is unclear and can potentially introduce a high degree of conservatism. If the less conservative LMI formulation, requiring the use of unstable multipliers, is used, the resultant BMI is of very high dimension due to the introduction of a Lyapunov inequality of the dimension of the closed-loop system to ensure closed-loop stability. In contrast, the robustness analysis results using a Riccati equation formulation easily extend to robust control design without placing any basis restrictions on the multipliers or introducing high dimensionality.

This research used a Riccati equation constraint to formulate fixed-architecture, robust control design methods that use general forms of the fixed-structure multipliers. The

proposed method relies on the development of an artificial cost function. This cost function also includes barrier functions to enforce positive definite constraints (e.g., on the Riccati solution P) which allows the constrained optimization problem (the constraints including $P>0$) to be converted into an unconstrained optimization problem. The cost function is not physically meaningful so we do not encounter the normal problems associated with making the barrier functions small as the last step of the optimization process. If the barrier terms are ignored and a certain term is added to the cost function, the cost function becomes an H_2 upper bound.

Due to the positive definite constraint on the Riccati solution, it is not possible to approach the solution to the optimization problem using standard descent methods. Hence, we develop probability-one homotopy algorithms to find the solution. This class of homotopy algorithms is distinct from classical continuation algorithms in that they follow the zero curve using the arc length parameter and not the actual homotopy parameter λ. This allows the zero curve to be nonmonotonic in λ and provides additional numerical robustness. In addition, the algorithms developed can be initialized with any stabilizing compensator and admissible multiplier, in contrast to previous algorithms.

3. A Comparison of Descent and Continuation Algorithms for H_2 Optimal, Reduced-Order Control Design

Journal Paper

E. G. Collins, Jr. and D. Sadhukhan, "A Comparison of Descent and Continuation Algorithms for H_2 Optimal Reduced-Order Control Design," submitted to the *International Journal of Control.*

Conference Paper

E. G. Collins, Jr. and D. Sadhukhan, "A Comparison of Descent and Continuation Algorithms for H_2 Optimal Reduced-Order Control Design," to be submitted to the *1997 American Control Conference.*

One of the deficiencies of modern control laws, developed by simply solving a pair of decoupled Riccati equations, in particular, linear-quadratic-guassian (LQG) control and full-order H_∞ control, is that the resultant control laws are always of the order of the design plant. These techniques, though relatively easy to implement computationally, do not allow the designer to constrain the architecture (e.g. order and degree of centralization) of the controller. Such constraints are often necessary in engineering practice due to throughput limitations of the control processors. Reduced-order control is therefore of paramount importance in practical control design. This research focuses on the design of H_2 optimal, reduced-order controllers.

Two main approaches have been developed to solve the H_2 optimal, reduced-order design problem. The first approach attempts to develop approximations to the optimal reduced-order controller by reducing the dimension of an LQG controller. These methods

are attractive because they require relatively little computation and should be used if possible. Unfortunately, they tend to yield controllers that either destabilize the system or have poor performance as the requested controller dimension is decreased or the requested control authority level is increased. Hence, if used in isolation, these methods do not yield a reliable methodology for reduced-order design. In addition, these methods do not extend to the design of decentralized controllers. However, it should be mentioned that, in regards to reduced-order control design, the indirect approaches at worst are valuable in providing good initial conditions for the direct approaches described below.

In contrast to controller reduction, direct approaches attempt to directly synthesize an optimal, reduced-order (or decentralized) controller by a numerical optimization scheme. There are two main classes of parameter optimization approaches to direct control design. The first class relies on the use of descent methods. Algorithms in this class reduce the H_2 cost at each iteration. The second class relies on the use of continuation methods. In contrast to the descent methods, the H_2 cost is not necessarily reduced at each iteration. It should be mentioned that continuation algorithms have also been developed to solve the "optimal projection equations," a set of four coupled Lyapunov and Riccati equations that characterize the H_2 optimal, reduced-order compensator. However, this approach was not be considered in this research.

From a practical design perspective it is important to determine which class of methods tends to be more numerically robust. As with the vast majority of numerical methods for nonconvex optimization problems, answers to these questions are extremely difficult to prove analytically. Instead, we must rely on numerical experimentation to observe trends. Hence, in this research the behavior of some standard descent methods (i.e., steepest descent, conjugate gradient, BFGS Quasi-Newton, and Newton's method) were compared to the corresponding behavior of the continuation algorithm of by considering design for three reduced-order control design problems appearing in the literature. The results clearly indicate that the continuation algorithm tends to be more numerically robust and is most efficient when the controller is constrained to a tridiagonal form.

4. Cost-Effective Parallel Processing for H_2/H_∞ Controller Synthesis

Journal Paper

Y. Ge., L. T. Watson, and E. G. Collins, Jr., "Cost-Effective Parallel Processing for H_2/H_∞ Controller Synthesis," submitted to the *International Journal of Control.*

Conference Paper

Y. Ge., L. T. Watson, and E. G. Collins, Jr., "Distributed Homotopy Algorithms for H_2/H_∞ Controller Synthesis," in *Parallel Processing for Scienticfic Computing,* P. E. Bjorstand, J. R. Gilbert, M. V. Mascagni, R. S. Schreiber, H. D. Simon, V. J. Torczor, and L. T. Watson, eds., SIAM, Philadephia, PA, pp. 84-89, 1995.

H_2/H_∞ mixed-norm controller synthesis is an important and interesting technique in modern control design which provides the means for simultaneously addressing H_2 and H_∞ performance objectives. In practice such controllers provide both nominal performance (via suboptimal H_2) and robust stability (via H_∞). Hence mixed-norm synthesis provides a technique for trading off performance and robustness, a fundamental objective in control design.

The H_2/H_∞ mixed-norm problem has been addressed in a variety of settings. One treatment utilized an H_2 cost bound as the basis for an auxiliary nonconvex constrained minimization problem, which is very difficult to solve without the global convergence of homotopy methods. A successful homotopy algorithm based on the Ly form parametrization has previously been developed.

The H_2/H_∞ control design algorithms can be used for controller design of systems such as the benchmark four disk system. This system is especially representative of the type of vibration control problems that arise in industrial problems involving rotating turbomachinery. H_2/H_∞ design can be used to develop controllers that are robust with respect to unmodeled dynamics and also guarantee a certain measure of nominal performance.

It should be mentioned that H_2/H_∞ theory has been used previously to develop complex structured singular value (CSSV) synthesis formalisms that *a priori* fix the structure of both the D-scales and the controller. Hence, an extension of the algorithms here will enable fixed-structure CSSV controller synthesis that blends H_2 and H_∞ performance objectives.

Practical applications often lead to large dense systems of nonlinear equations which are time-consuming to solve on a serial computer. For these systems, parallel processing may be the only feasible means to achieving solution algorithms with acceptable speed. One economical way of achieving parallelism is to utilize the aggregate power of a network of heterogeneous serial computers. In industrial environments where interactive design is often the practice, the parallel code can be easily incorporated into interactive software such as MATLAB or Mathematica with proper setup of the network computers. To the engineering users the design environment is identical. However, the computations are faster.

The most expensive part of the previously developed H_2/H_∞ homotopy algorithm is the computation of the Jacobian matrix, which can be parallelized easily to run across an Ethernet network with little modification of the original sequential code, and which has relatively large task granularity. There is a trade-off between the programming effort and the speedup of the parallel program. To obtain a better speedup, other parts of the homotopy algorithm, such as finding the solution to the Riccati equations and the QR factorization to compute the kernel of the Jacobian matrix, need to be parallelized as well.

In this sutdy the homotopy algorithm for H_2/H_∞ controller synthesis was parallelized to run on a network of workstations using PVM (Parallel Virtual Machine) and on an Intel Paragon parallel computer, under the philosophy that as few changes as possible are to be made to the sequential code while achieving an acceptable speedup. The parallelized computation is that of the Jacobian matrix, which is carried out in the master-slave paradigm by functional parallelism; that is, each machine computes a different column of the Jacobian matrix with its own data. Unless the Riccati equation solver is parallelized, there is a large amount of data needed for each slave process at each step of the homotopy algorithm. To avoid sending too many large messages across the network or among different nodes on the Intel Paragon, all slave processes repeat part of the computation done by the master process, which therefore decreases the speedup of the parallel computation.

The speedups of the parallel code were compared as the number of workstations increases or as the number of nodes increases on an Intel Paragon and as the size of the problem varies. A reasonable speedup can be achieved using an existing network of workstations compared to that of using an expensive parallel machine, the Intel Paragon. It was demonstrated that for a large problem, the approach of using a network of workstations to parallelism is feasible and practical, and provides an efficient and economical computational method to parallelize a homotopy based algorithm for H_2/H_∞ controller synthesis in a workstation-based interactive design environment.

5. An Object-oriented Approach to Semidefinite Programming

Journal Paper

Y. Ge, L. T. Watson, and E. G. Collins, Jr., "An Object-oriented Approach to Semidefinite Programming," submitted to the *International Journal of Computer Mathematics*.

Conference Paper

Y. Ge, L. T. Watson, and E. G. Collins, Jr., "An Object-oriented Approach to Semidefinite Programming," to be submitted to the *1997 American Control Conference*.

Object-oriented design and programming has been a major theme in software engineering in recent years. Traditional design or classical design, which has been the main software design paradigm until about the mid 80's, concentrates on the actions that a system has to take and decomposes the sytem into separte units or modules according to their functionalities. In object-oriented design a system to be modeled is viewed as a collection of objects, each of which has its own attributes and the operations performed on an object or functions acting on an object are also defined in one syntactic unit. Objects communicate by passing messages or by calling functions from other objects which provide services. Object-oriented design is developing an object-oriented model of a system and can be realized (implemented) by object-oriented programming using languages such as C++, FORTRAN 90, or Smalltalk.

The advantages of object-oriented design and programming have been described widely elsewhere. A short summary is provided here. First, an object is an independent entity that is encapsulated in one syntactic unit. The definition of an object consists of the definition of the attributes of the object along with operations that can be performed on the object and the services or function calls provided by the object. Encapsulation hides the implementation details of an object and makes the program easier to modify. Any subsequent changes to the program can be localized, making the resulting program more easily maintained.

The second advantage is information hiding. Definitions of an object which need not be known to other objects are inaccessible to other objects, preventing them from being changed accidentally. In other works, information hiding makes implementation details of an object inaccessible to other objects. However, the designer has the freedom to decide what to hide and what not to hide.

The third advantage is code reuse. Inheritance enables the definition of a new object, which can be viewed as a subclass of an existing object, without having to repeat some of the details. The new object can inherit attributes or operations from its ancestor. Inheritance is one way to support reuse of existing objects. There are different kinds of reuse in object-oriented programming; inheritance is only one of them.

One of the most popular object-oriented programming languages is C++, which was used to implement the algorithm of this research. Some of the reasons why C++ is so widely used are upward compatibility with *C*, design emphasis on efficiency and performance, and the availability of many useful libraries and tools. For instance, the Gnu C++ compiler and other tools are available on a wide range of platforms and provide good performance, programming environments, and reasonable compliance with ANSI standards.

There are many available libraries such as IML++, SparseLib++, STL, and others which emphasize numerical computation. One notable package is LAPACK++, developed by Dongarra et al., which is a C++ interface to LAPACK and BLAS. It has been shown that performance of programs using the package is comparable to calling LAPACK and BLAS directly, anc can at the same time reap benefits of object-oriented programming.

This research developed and analyzed object-oriented design and implementation of an algorithm for semidefinite programming. Semidefinite programming refers to minimizing a linear function subject to a linear matrix inequality. That is

$$\begin{aligned} &\underset{x \in \boldsymbol{R}^m}{\text{minimize}}\ c^T x \\ &\text{subject to } F(x) \geq 0, \end{aligned}$$

where

$$F(x) \equiv F_0 + \sum_{i=1}^{m} x_i F_i,$$

$c \in \boldsymbol{R}^m$, and $F_0, \ldots, F_m \in \boldsymbol{R}^{n \times n}$ are symmetric matrices. $F(x) \geq 0$ means that $F(x)$ is positive semidefinite.

Many problems in controls engineering can be cast in terms of a semidefinite programming problem. Since a semidefinite programming problem is a convex optimization problem, which can be solved by interior point methods, it has attracted the attention of many researchers in interior point methods. There is a *C* implementation of a primal-dual algorithm for the semidefinite problem. A C++ implementation of that primal-dual algorithm was developed in this resarch to explore the possible benefits of object-oriented design. Because of the similarity of the primal-dual algorithm with other interior point algorithms for solving the semidefinite programming problem, the design and implementation methodology developed in this research can be easily modified and applied to other interior point algorithms.

The performance of a C++ implementation of the primal-dual algorithms for semidefinite programming is compared with the existing *C* implementation. While the CPU times of the two implementations are comparable to each other, the C++ version offers the advantages mentioned earlier in this section. Segments of the code are used to illustrate object-oriented features of the implementation.

Appendix A

"Probability-One Homotopy Algorithms for Robust Controller Synthesis with Fixed-Structure Multipliers"

October 10, 1995
revised April 9, 1996

Probability-One Homotopy Algorithms for Robust Controller Synthesis with Fixed-Structure Multipliers

by

Emmanuel G. Collins, Jr.
Dept. of Mechanical Engineering
Florida A&M - Florida State
2525 Pottsdamer Street
Tallahassee, FL 32310-6046
ecollins@eng.fsu.edu

Wassim M. Haddad
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0150
wm.haddad@aerospace.gatech.edu

Layne T. Watson
Departments of Computer
Science and Mathematics
Virginia Polytechnic Institute
and State University
Blacksburg, VA 24061-0106
ltw@cs.vt.edu

Debashis Sadhukhan
Dept. of Mechanical Engineering
Florida A&M - Florida State
2525 Pottsdamer Street
Tallahassee, FL 32310-6046
sadhukhan@eng.fsu.edu

[1]This research was supported in part by the Air Force Office of Scientific Research under Grant F49620-95-1-0244 and the National Science Foundation under Grant ECS-9496249

Summary

Continuation algorithms that avoid multiplier-controller iteration have been developed earlier for fixed-architecture, mixed structured singular value controller synthesis. These algorithms have only been formulated for the special case of Popov multipliers and rely on an *ad hoc* initialization scheme. In addition, the algorithms have not used the prediction capabilities obtained by computing the Jacobian matrix of the continuation (or homotopy) map, and have assumed that the homotopy zero curve is monotonic. This paper develops probability-one homotopy algorithms based on the use of general fixed-structure multipliers. These algorithms can be initialized using an arbitrary (admissible) multiplier and a stabilizing compensator. In addition, as with all probability-one algorithms, the homotopy zero curve is not assumed to be monotonic and prediction is accomplished by using the homotopy Jacobian matrix. This approach also appears to have some advantages over the bilinear matrix inequality (BMI) approaches resulting from extensions of the LMI framework for robustness analysis.

Keywords: Fixed-structure multipliers, multivariable stability margin, LMI's, BMI's, homotopy algorithms

Running Title: Robust Controller Synthesis with Fixed-Structure Multipliers

1. Introduction

During the past two decades, major advancements have been made in robust control theory. Building upon H_∞ theory, the structured singular value (SSV)[1, 2] was defined as a nonconservative robustness measure for the analysis of linear systems with dynamic, arbitrary phase, multiple-block uncertainty. The supremum of the structured singular value over nonnegative frequencies is the inverse of the multivariable stability margin (see References 3, 4 and the references therein). The initial developments in structured singular value theory focussed on dynamic uncertainty with arbitrary phase (often called "complex uncertainty") and hence, although less conservative than H_∞ theory, could still yield very conservative robustness bounds for systems with parametric uncertainty. This led to the development of mixed (i.e., real and complex) structured singular value (MSSV) theory[5, 6] which considers block-diagonal uncertainty with both dynamic and real scalar parametric elements.

Parallel research addressed the issue of real parameter uncertainty using absolute stability theory such as Popov analysis[7-12] and was developed by recognizing the relationship between sector bounded nonlinearities and interval bounds on linear uncertainties. This work was soon seen to provide an upper bound for the MSSV.[8, 10, 13] In fact, in contrast to the initial work on the MSSV, this research provided the first *fixed-structure* multiplier versions of MSSV theory. A unique contribution of some of this work is that it led to the development of upper bounds on an H_2 cost functional over the uncertainty set under consideration. By optimizing this upper bound and using a Riccati equation constraint, continuation algorithms have been developed for MSSV controller synthesis.[14-16] A related algorithm for complex stuctured singular value (CSSV) controller synthesis is given in Reference 17. Note that the H_2 approach allows the direct design of fixed-architecture (e.g., reduced-order or decentralized) controllers and the simultaneous optimization of the controller and (fixed-structure) multipliers, hence avoiding M-K (i.e., multiplier-controller) iteration schemes. However, to date the synthesis algorithms have been formulated only for the case of the the Popov multiplier. In addition, the algorithms rely on an *ad hoc* initialization scheme, have not used the prediction capabilities obtained by computing the Jacobian matrix of the homotopy (or continuation) map, and have assumed that the homotopy curve is monotonic.

A similar line of research has been developed independently in Reference 18-20. This work also provides a fixed-structure multiplier version of the MSSV but, unlike the approach described in References 7-10,12, this approach develops multipliers for strictly linear uncertainties. The associated robustness analysis was originally formulated in terms of a convex, frequency-domain optimization problem but has recently been reformulated in terms of a (convex) linear-matrix-inequality

(LMI) problem.[19, 21] These results have led to the recognition that robust control design can be approached via solving a (nonconvex) "bilinear matrix inequality" (BMI).[22-24] This approach allows the design of fixed-architecture controllers and can be implemented without using M-K iteration. To obtain a reasonably sized BMI, the multiplier set must be restricted to lie in the span of a stable basis.[22] However, the choice of this basis is unclear and can potentially introduce a high degree of conservatism. If the less conservative LMI formulation, requiring the use of unstable multipliers, is used, the resultant BMI is of very high dimension due to the introduction of a Lyapunov inequality of the dimension of the closed-loop system to ensure closed-loop stability.[24] In contrast, the robustness analysis results using a Riccati equation formulation easily extend to robust control design without placing any basis restrictions on the multipliers or introducing high dimensionality.

This paper uses a Riccati equation constraint to formulate fixed-architecture, robust control design methods that use general forms of the fixed-structure multipliers. The proposed method relies on the development of an artificial cost function. This cost function also includes barrier functions to enforce positive definite constraints (e.g., on the Riccati solution P) which allows the constrained optimization problem (the constraints including $P > 0$) to be converted into an unconstrained optimization problem. The cost function is not physically meaningful so we do not encounter the normal problems associated with making the barrier functions small at the last step of the optimization process. (See Reference 25 for a discussion of this negative feature of standard barrier function methods.) If the barrier terms are ignored and a certain term is added to the cost function, the cost function becomes an H_2 upper bound.

Due to the positive definite constraint on the Riccati solution, it is not possible to approach the solution to the optimization problem using standard descent methods. Hence, we develop probability-one homotopy algorithms[26, 27] to find the solution. This class of homotopy algorithms is distinct from classical continuation algorithms[28] in that they follow the zero curve using the arc length parameter and not the actual homotopy parameter λ. This allows the zero curve to be nonmonotonic in λ and provides additional numerical robustness. In addition, the algorithms developed here can be initialized with any stabilizing compensator and admissible multiplier, in contrast to the algorithms of References 14-17.

1.1. Notation and Definitions

Let $\mathcal{R}$ and $\mathcal{C}$ denote the real and complex numbers, $\mathcal{R}^{m\times m}$ and $\mathcal{C}^{m\times m}$ the real and complex $m \times m$ matrices, let $(\cdot)^T$ and $(\cdot)^*$ denote transpose and complex conjugate transpose, let "Re" and "Im" denote real and imaginary part, and let I_n or I denote the $n \times n$ identity matrix. Furthermore

we write $\sigma_{\max}(\cdot)$ for the maximum singular value, "tr" for the trace operator, and $M \geq 0$ ($M > 0$) to denote the fact that the Hermitian matrix M is nonnegative (positive) definite. The Hermitian and skew-Hermitian parts of an arbitrary complex square matrix G are defined by He $G \triangleq \frac{1}{2}(G + G^*)$ and Sh $G \triangleq \frac{1}{2}(G - G^*)$, respectively. Finally, vec$(\cdot)$ denotes the standard column stacking operator.

Next, we establish certain key definitions used later in the paper. Let $n(s)$ and $d(s)$ be polynomials in s with real coefficients. A function $g(s)$ of the form $g(s) = n(s)/d(s)$ is called a *real rational function.* The function $g(s)$ is called *proper* (resp., *strictly proper*) if deg $n(s) \leq$ deg $d(s)$ (resp., deg $n(s) <$ deg $d(s)$), where "deg" denotes the degree of the respective polynomials. A *real-rational matrix function* is a matrix whose elements are rational functions with real coefficients. Furthermore, a *transfer function* $G(s)$ is called *proper* (resp., *strictly proper*) if every element of $G(s)$ is proper (resp., strictly proper). In this paper we assume all transfer functions are real-rational matrix functions. Also, we define $G^*(s) \triangleq G^T(-s)$ for transfer functions $G(s)$.

An *asymptotically stable transfer function* is a transfer function each of whose poles is in the open left half plane. Finally, a *Lyapunov stable transfer function* is a transfer function each of whose poles is in the closed left half plane with semi-simple poles on the imaginary axis. Let $G(s) \sim \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ denote a state space realization of a transfer function $G(s)$, that is, $G(s) = C(sI - A)^{-1}B + D$.

A square transfer function $G(s)$ is called *positive real* (resp., *generalized positive real*)[29] if $G(s)$ is Lyapunov stable and He $G(s)$ is nonnegative definite for Re$[s] > 0$ (resp., $G(s)$ has no imaginary poles and He $G(j\omega)$ is nonnegative definite for all $\omega \in \mathcal{R}$). A square transfer function is called *strictly positive real* [30] (resp., *strictly generalized positive real*) if $G(s)$ is asymptotically stable and He $G(j\omega)$ is positive definite for $\omega \in \mathcal{R}$ (resp., $G(s)$ has no imaginary poles and He $G(j\omega)$ is positive definite for $\omega \in \mathcal{R}$). A square transfer function $G(s)$ is *strongly positive real* (resp., *strongly generalized positive real*) if it is strictly positive real (resp., strictly generalized positive real) and $D + D^T > 0$, where $D \triangleq G(\infty)$. Note that although a minimal realization of a positive real transfer function is stable in the sense of Lyapunov, a minimal realization of a generalized positive real transfer function may be unstable.

1.2. Paper Organization

Section 2 presents the general framework for robustness analysis with fixed-structure multipliers and briefly discusses the BMI approach to fixed-architecture, robust control synthesis. Section 3 demonstrates that an alternative formulation to robust synthesis is in terms of a Riccati equation feasibility problem. Section 4 develops probability-one homotopy algorithms to solve the Riccati equation feasibility problem. Section 5 specializes the results to the special case of the Popov

multiplier and presents a numerical example. Finally, Section 6 presents conclusions and directions for further research.

2. Multiplier Methods in Robust Analysis

In this section we review the framework for mixed uncertainty robustness analysis with fixed-structure multipliers. The exposition generally follows that presented in References 12,19-21. We begin by considering the standard uncertainty feedback configuration of Figure 1. where $G(s) \in \mathcal{C}^{m\times m}$ is asymptotically stable and $G(s) \sim \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$. It is assumed that the uncertainty $\Delta \in \mathcal{C}^{m\times m}$ belongs to the set

$$\mathbf{\Delta}_\gamma \triangleq \{\Delta = \text{block-diag}(\Delta_1, ..., \Delta_p) : \Delta_i \in \mathcal{I}_i,\ \sigma_{\max}(\Delta_i) \leq \gamma, i = 1, ..., p,\ \sum_{i=1}^{p} k_i = m\}, \qquad (2.1)$$

where $\mathcal{I}_i \subseteq \mathcal{C}^{k_i\times k_i}$ denotes the internal structure of the uncertainty block Δ_i and $\gamma > 0$. For example $\mathcal{I}_i$ may be given by any of the following five sets:

$$\mathbf{\Delta}^{\mathrm{I}} \triangleq \mathcal{C}^{k_i\times k_i}, \qquad (2.2)$$

$$\mathbf{\Delta}^{\mathrm{II}} \triangleq \{\Delta_i = \delta_i I_{k_i} : \delta_i \in \mathcal{C}\}, \qquad (2.3)$$

$$\mathbf{\Delta}^{\mathrm{III}} \triangleq \{\Delta_i = \delta_i I_{k_i} : \delta_i \in \mathcal{R}\}, \qquad (2.4)$$

$$\mathbf{\Delta}^{\mathrm{IV}} \triangleq \{\Delta_i \in \mathcal{R}^{k_i\times k_i} : \Delta_i = \Delta_i^T\}, \qquad (2.5)$$

$$\mathbf{\Delta}^{\mathrm{V}} \triangleq \{\Delta_i = \left[\begin{array}{cc} -\delta_\nu & \delta_\omega \\ -\delta_\omega & -\delta_\nu \end{array}\right] : \delta_\nu, \delta_\omega \in \mathcal{R}\}. \qquad (2.6)$$

Note that $\mathbf{\Delta}^{\mathrm{I}}$, $\mathbf{\Delta}^{\mathrm{II}}$, and $\mathbf{\Delta}^{\mathrm{III}}$ are standard in the literature, corresponding respectively to complex matrix block uncertainty, repeated complex scalar uncertainty, and repeated real scalar uncertainty. $\mathbf{\Delta}^{\mathrm{IV}}$ is symmetric, real matrix block uncertainty, while $\mathbf{\Delta}^{\mathrm{V}}$ can be used to describe uncertainty in the imaginary and real parts of a structural system represented in real normal form. If the uncertainty is of the form described by $\mathbf{\Delta}^{\mathrm{IV}}$ or $\mathbf{\Delta}^{\mathrm{V}}$, it is possible to represent the uncertainty by $\mathbf{\Delta}^{\mathrm{III}}$. However, as discussed in Reference 12 this reformulation leads to increased conservatism and numerical complexity. The ensuing discussion is not restricted to these forms of uncertainty, but they are important special cases and will be used to provide concrete illustrations of the subsequent concepts.

To state the multivariable absolute stability criterion for $\Delta \in \mathbf{\Delta}_\gamma$ we define the sets of Hermitian, frequency-dependent, scaling matrix functions by

$$\mathcal{D}_i \triangleq \{D_i : j\mathcal{R} \cup \infty \to \mathcal{C}^{k_i \times k_i} : D_i(j\omega) \geq 0, \\ D_i(j\omega)\Delta_i = \Delta_i D_i(j\omega), \ \omega \in \mathcal{R}, \ \Delta_i \in \mathcal{I}_i, \ i = 1, ..., p\}, \tag{2.7}$$

$$\mathcal{N}_i \triangleq \{N_i : j\mathcal{R} \cup \infty \to \mathcal{C}^{k_i \times k_i} : N_i(j\omega) = N_i^*(j\omega), \\ N_i(j\omega)\Delta_i = \Delta_i^* N_i(j\omega), \ \omega \in \mathcal{R}, \ \Delta_i \in \mathcal{I}_i, \ i = 1, ..., p\}. \tag{2.8}$$

Furthermore, define the sets $\mathcal{M}_i$ and $\mathcal{M}$ of multiplier transfer functions by

$$\mathcal{M}_i \triangleq \{M_i(s) = D_i(s) + Q_i(s) : D_i(j\omega) \in \mathcal{D}_i, \ Q_i(j\omega) = j\omega N_i(j\omega), \ N_i(j\omega) \in \mathcal{N}_i, \ i = 1, ..., p\}, \tag{2.9}$$

$$\mathcal{M} \triangleq \{M(s) \in \mathcal{C}^{m \times m} : M(s) = \text{block-diag}(M_1(s), ..., M_p(s)), \ M_i(s) \in \mathcal{M}_i, \ i = 1, ..., p\}. \tag{2.10}$$

Note that in (2.9), $D_i(j\omega) = \text{He}\, M_i(j\omega) \geq 0$ and $N_i(j\omega) = -j\text{Sh}\, M_i(j\omega)$. Furthermore, $M(s) \in \mathcal{M}$ satisfies

$$\text{He}\, M(j\omega) \geq 0, \ \ \omega \in \mathcal{R} \cup \infty, \tag{2.11}$$

and is not necessarily stable.

Theorem 1. Suppose that $G(s)[I - \gamma G(s)]^{-1}$ is asymptotically stable. If there exists $M(s) \in \mathcal{M}$ such that

$$\text{He}[M(j\omega)T_\gamma(j\omega)] > 0, \ \omega \in \mathcal{R} \cup \infty, \tag{2.12}$$

where

$$T_\gamma(s) \triangleq [I + \gamma G(s)][I - \gamma G(s)]^{-1}, \tag{2.13}$$

then the negative feedback interconnection of $G(s)$ and Δ is asymptotically stable (or, equivalently, $\det(I + G(j\omega)\Delta) \neq 0$, $\omega \in \mathcal{R}$) for all $\Delta \in \mathbf{\Delta}_\gamma$.

Proof. A rigorous proof of this result is given in Reference 12. Similar results are presented in References 19-21. $\triangle$

Remark 1. Note that[21]

$$T_\gamma(s) \sim \left[\begin{array}{c|c} A + \gamma B(I - \gamma D)^{-1}C & \sqrt{2\gamma}B(I - \gamma D)^{-1} \\ \hline \sqrt{2\gamma}(I - \gamma D)^{-1}C & (I + \gamma D)(I - \gamma D)^{-1} \end{array}\right]. \tag{2.14}$$

Using the coprime factorization result presented in Reference 31, it follows that $M(s)$ can be factored as

$$M(s) = [M_{\mathrm{B}}^*(s)]^{-1} M_{\mathrm{A}}(s), \tag{2.15}$$

where both $M_{\mathrm{A}}(s)$ and $M_{\mathrm{B}}(s)$ are asymptotically stable and nonunique. In practice, stable $M_{\mathrm{A}}(s)$ and $M_{\mathrm{B}}(s)$ satisfying (2.15) can be computed using the approach pioneered in Reference 32 which is also given in Reference 31. In Reference 20 (2.15) is used to prove the following important corollary to Theorem 1.

Corollary 1. Assume $M(s) \in \mathcal{M}$, and $M_{\mathrm{A}}(s)$ and $M_{\mathrm{B}}(s)$ are asymptotically stable transfer function matrices satisfying (2.15). Then (2.12) in Theorem 1 holds if and only if

$$\mathrm{He}[M_{\mathrm{A}}(j\omega) T_\gamma(j\omega) M_{\mathrm{B}}(j\omega)] > 0, \ \omega \in \mathcal{R} \cup \infty. \tag{2.16}$$

Corollary 1 allows us to develop robust controllers based on positive real theory. Such a method is developed in Section 3.

We now characterize $\mathcal{M}_i$ for $\mathcal{I}_i$ equal to the sets defined by (2.2)-(2.6). These characterizations are restatements of results given in Reference 12. Multiplier sets corresponding to $\mathbf{\Delta}^{\mathrm{I}}$, $\mathbf{\Delta}^{\mathrm{II}}$, and $\mathbf{\Delta}^{\mathrm{III}}$ are given in References 19, 20. The set corresponding to $\mathbf{\Delta}^{\mathrm{II}}$ [20] differs from that given here. The following characterizations are useful in constructing state space realizations of the multipliers. In particular, for $\mathcal{I}_i = \mathbf{\Delta}^{\mathrm{I}}$

$$\mathcal{M}_i = \{m_i(s) I_{k_i} : \mathrm{Re}[m_i(j\omega)] \geq 0, \ \mathrm{Im}[m_i(j\omega)] = 0, \ \omega \in \mathcal{R} \cup \infty\}, \tag{2.17}$$

while for $\mathcal{I}_i = \mathbf{\Delta}^{\mathrm{II}}$

$$\mathcal{M}_i = \left\{M_i(s) \in \mathcal{C}^{k_i \times k_i} : \mathrm{He}\, M_i(j\omega) \geq 0, \ M(j\omega) = M^*(j\omega), \ \omega \in \mathcal{R} \cup \infty\right\}. \tag{2.18}$$

Note that if we denote $\mathcal{M}_i(s) = [m_{jk}(s)]_{(j,k=1,\ldots,k_i)}$, $M(j\omega) = M^*(j\omega)$ implies

$$\mathrm{Im}[m_{jj}(j\omega)] = 0, \quad m_{jk}(j\omega) = m_{kj}(-j\omega), \ j \neq k.$$

Furthermore, for $\mathcal{I}_i = \mathbf{\Delta}^{\mathrm{III}}$

$$\mathcal{M}_i = \left\{M_i(s) \in \mathcal{C}^{k_i \times k_i} : \mathrm{He}\, M(j\omega) \geq 0, \ \omega \in \mathcal{R} \cup \infty\right\}, \tag{2.19}$$

while for $\mathcal{I}_i = \mathbf{\Delta}^{\mathrm{IV}}$

$$\mathcal{M}_i = \{m_i(s) \mathcal{I}_{k_i} : m_i(s) \in \mathcal{C}, \ \mathrm{Re}[m_i(j\omega)] \geq 0, \ \omega \in \mathcal{R} \cup \infty\}. \tag{2.20}$$

Finally, for $\mathcal{I}_i = \mathbf{\Delta}^{\mathrm{V}}$

$$\mathcal{M}_i = \{M_i(s) = D(s) + Q(s) : D(s) = \begin{bmatrix} d_{11}(s) & d_{12}(s) \\ -d_{12}(s) & d_{11}(s) \end{bmatrix},$$
$$Q(s) = \begin{bmatrix} q_{11}(s) & q_{12}(s) \\ q_{12}(s) & -q_{11}(s) \end{bmatrix}, \mathrm{He}\, D(j\omega) > 0,$$
$$\mathrm{Re}\, Q(j\omega) = 0, \mathrm{Im}[d_{11}(j\omega)] = \mathrm{Re}[d_{12}(j\omega)] = 0,\ \omega \in \mathcal{R} \cup \infty\}. \tag{2.21}$$

2.1. The Structured Singular Value and Robust Performance

For a multiple block-structured uncertainty set $\mathcal{I}$, with possibly repeated scalar elements, complex scalar elements, real blocks, and complex blocks,[12] the structured singular value of a complex matrix $G(j\omega)$ is defined by

$$\mu(G(j\omega)) \triangleq \left(\inf_{\Delta \in \mathcal{I}} \{\sigma_{\max}(\Delta) : \det(I + G(j\omega)\Delta) = 0\} \right)^{-1},$$

where by convention $\mu(G(j\omega)) = 0$ if there does not exist $\Delta \in \mathcal{I}$ such that $\det(I + G(j\omega)\Delta) = 0$. The structured singular value nonconservatively characterizes the robust stability of the uncertainty feedback system of Figure 1, as stated by the following theorem.[2, 12]

Theorem 2. Suppose $G(s)$ is asymptotically stable. Then the negative feedback interconnection of $G(s)$ and Δ is asymptotically stable for all $\Delta \in \mathbf{\Delta}_\gamma$ if and only if

$$\mu(G(j\omega)) < \gamma^{-1},\ \omega \in \mathcal{R} \cup \infty. \tag{2.22}$$

Remark 2. The parameter $\gamma_{\mathrm{m}} \triangleq 1/\sup_{\omega \geq 0} \mu(G(j\omega))$ is the multivariable stability margin.[4]

Next define

$$\mu_{\mathrm{abs}}(G(j\omega)) \triangleq \inf\{\gamma > 0 : \text{ there exists } M(\cdot) \in \mathcal{M} \text{ such that } \mathrm{He}[M(j\omega)T_\gamma(j\omega)] > 0,\ \omega \in \mathcal{R} \cup \infty\}. \tag{2.23}$$

Then the following holds.

Theorem 3. For $\omega \in \mathcal{R} \cup \infty$, let $G(j\omega)$ be a complex matrix. Then

$$\mu(G(j\omega)) \leq \mu_{\mathrm{abs}}(G(j\omega)).$$

Proof A rigorous proof is given in Reference 12. Similar results are stated in References 19, 20. $\triangle$

The significance of the above theorem is that it allows us to consider both robust stability and robust performance in the same setting.[6] Hence, as was proved for structured singular value analysis with purely complex uncertainty,[2] robust performance can be ensured by appropriate inclusion of a "fictitious" full complex uncertainty block.

2.2. Linear Matrix Inequality and Riccati Equation Characterizations of (Generalized) Positive Real Matrix Functions

Theorem 1 characterizes robustness in terms of the strictly generalized positive real condition (2.12) while Corollary 1 relies on the strictly positive real condition (2.16). Hence, to implement the robustness test of Theorem 1 using state space computations requires state space characterizations of strictly generalized positive real and strictly positive real transfer functions.

State space conditions for *strictly* positive real transfer functions are given in Reference 33, but include observability and rank conditions which are difficult to incorporate into numerical schemes. Hence, the lemma below provides state space characterizations of strongly generalized positive real matrices and strongly positive real matrices, special cases respectively of strictly generalized positive real matices and strictly positive real matrices.

Lemma 1. Let $G(s)$ be square with $G(s) \sim \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ where (A, B) is controllable. Then the following statements are equivalent:

1. $G(s)$ is strongly generalized positive real.

2. There exist $\epsilon > 0$ and symmetric P such that

$$\begin{bmatrix} -A^TP - PA & -PB + C^T \\ -B^TP + C & (D - \epsilon I) + (D - \epsilon I)^T \end{bmatrix} \geq 0.$$

Furthermore, if (1) and (2) hold, then:

4. (A, C) is observable if and only if P is nonsingular.

5. $G(s)$ is strongly positive real if and only if A is asymptotically stable and $P \geq 0$. In this case, (A, C) is observable if and only if $P > 0$.

6. If (A, C) is observable, $G(s)$ is strongly positive real if and only if $D + D^T > 0$ and there exist $P > 0$ and $\epsilon > 0$ such that

$$0 = A^TP + PA + (B^TP - C)^T(D + D^T)^{-1}(B^TP - C) + \epsilon I. \tag{2.24}$$

Proof. The equivalence of statements 1 and 2 is shown in Reference 19. Statements 3 and 4 are proved in Reference 29 and statement 5 is proved in Reference 9. $\triangle$

Lemma 1 shows that strongly generalized positive reality and generalized positive reality is equivalent to the existence of a certain solution to an LMI. Furthermore, the latter condition (under the assumption that (A, B, C, D) is a minimal state space realization) is equivalent to the existence of a certain solution to a Riccati equation. This Riccati equation is subsequently used to develop probability-one homotopy maps for robust controller synthesis with fixed-structure multipliers.

2.3. Bilinear Matrix Inequality Approaches to Robust Controller Synthesis

Both References 19, 21 describe ways of using the LMI's corresponding to generalized positive real tests to develop robustness tests expressed in terms of the existence of a solution to an LMI. The key is to represent the multiplier $M(s)$ in the strictly generalized positive real test (2.12) of Theorem 1 in such a way that its parameters appear linearly in the corresponding state space test of Lemma 1. When the LMI robustness *analysis* results are generalized to fixed-architecture, robust control *synthesis*, a BMI results.

A straightforward way of achieving the desired representation of $M(s)$ is described in Reference 21 and is based on earlier ideas given in Reference 20. This approach requires expressing the multiplier $M(s)$ in terms of a basis expansion. In particular,

$$M(s) = \sum_{j=1}^{r} \beta_j \bar{M}^{(j)}(s), \tag{2.25}$$

where $\beta_j \geq 0$ and $\bar{M}^j(s) \in \mathcal{M}$, $j = 1, ..., r$. A substantial weakness of this approach is the difficulty of choosing the basis $\{\bar{M}^{(1)}(s), ..., \bar{M}^{(r)}(s)\}$. Specifically, for a given multiplier $M(s) \in \mathcal{M}$ and a given basis, the approach in Reference 21 does not guarantee that there exists nonnegative β_j such that $M(s)$ is given by (2.25).

The approach proposed in Reference 19 does not suffer from these weaknesses. It is based on the following result.

Lemma 2. [20] Equation (2.12) is satisfied for some transfer matrix $M^{(0)}(s) \in \mathcal{M}$ if and only if there exist a real polynomial matrix $M(s) \in \mathcal{M}$ for which (2.12) holds. Furthermore, if $M^{(0)}(s)$ is factored as $M^{(0)}(s) = \frac{1}{d^{(0)}(s)} N^{(0)}(s)$ where $N^{(0)}(s)$ is a real polynomial matrix and $d^{(0)}(s)$ is a scalar-valued real polynomial, then the degree of $M(s)$ need not be greater than the sum of degrees of $N^{(0)}(s)$ and $d^{(0)}(s)$, and in fact one can choose $M(s) = d^{(0)}(-s)N^{(0)}(s)$. In addition, if $M(s)$ is of order $2n$, then for all n^{th}-order real polynomials $d(s)$ having no zeros on the $j\omega$ axis,

$\tilde{M}(s) = \frac{M(s)}{d(-s)d(s)} \in \mathcal{M}$ and satisfies (2.12).

Remark 3. Lemma 2 allows one to restrict the multiplier search to $2n^{\text{th}}$ order, real polynomial matrices $M(s)$. To obtain state space realizable transfer functions we can consider

$$\tilde{M}(s) = \frac{M(s)}{d(s)d(-s)}, \tag{2.26}$$

where $d(s)$ is an arbitrary n^{th}-order polynomial having no zeros on the $j\omega$-axis.

Remark 4. Note that since the zeros of $d(-s)$ are the mirror images of the zeros of $d(s)$ about the imaginary axis, $\tilde{M}(s)$ given by (2.26) is *always* an unstable multiplier.

This latter approach is powerful but as eluded to in Remark 4 always produces an unstable multiplier. When extended to fixed architecture, robust control design, this approach results in a BMI with very high dimension since it requires the introduction of a Lyapunov inequality of dimension equal to that of the closed-loop system to ensure closed-loop stability. We hence begin the development of an alternative scheme without these limitations.

3. A Riccati Equation Approach to Robust Controller Synthesis with Fixed-Structure Multipliers

We now give exclusive attention to robustness tests that are expressed in terms of positive real conditions, as opposed to generalized positive real conditions. Hence, we will consider robustness tests corresponding to Corollary 1. The developments here differ dramatically from those in the previous section since we use the Riccati equation characterization of strong positive reality given in Lemma 1 instead of the LMI characterization of strong generalized positive reality which is also given in Lemma 1.

We focus on the uncertainty structures corresponding to the sets $\mathbf{\Delta}^{\text{I}}$ and $\mathbf{\Delta}^{\text{III}}$, defined respectively by (2.2) and (2.4). Hence, we develop fixed-structure multiplier tests for the complex, block-structured uncertainty considered by classical complex structured singular value analysis [1, 2] and the real, diagonal uncertainty considered by classical real structured singular value analysis.[5, 6] Although not detailed here, the uncertainty structures corresponding to the sets $\mathbf{\Delta}^{\text{II}}$, $\mathbf{\Delta}^{\text{IV}}$, and $\mathbf{\Delta}^{\text{V}}$, may also be considered in the framework of this section. In addition, the results may be extended in a straightforward manner to mixed uncertainty sets.

The resulting numerical algorithms are probability-one homotopy algorithms. A detailed discussion of the numerical algorithms is reserved for the next section. Here, we lay the necessary

foundation for these algorithms. We begin by developing a constructive characterization of multiplier factors $M_A(s)$ and $M_B(s)$ corresponding to complex, block-structured uncertainty and real, diagonal uncertainty. These constructions are essential to the subsequent developments.

3.1. Complex, Block-Structured Uncertainty

From (2.10) and (2.17) it follows that a multiplier corresponding to complex block-diagonal uncertainty is given by

$$M(s) = \text{block-diag}\left(m_1(s)I_{k_1}, ..., m_p(s)I_{k_p}\right), \tag{3.1}$$

$$\text{Re } m_i(j\omega) > 0,\ \omega \in \mathcal{R} \cup \infty, \tag{3.2}$$

$$\text{Im } m_i(j\omega) = 0.\ \omega \in \mathcal{R} \cup \infty. \tag{3.3}$$

Note that we have replaced the weak inequality in (2.17) with a strict inequality in (3.2).

Lemma 3. For uncertainty structures corresponding to (2.2) there exists $\tilde{M}(s)$ with no zeros or poles on the imaginary axis and satisfying the compatibility conditions (3.1)-(3.3) and the robustness test (2.12), if and only if there exists $M(s)$ satisfying (2.12) and (3.1) such that

$$M(s) = \bar{M}(-s)\bar{M}(s), \tag{3.4}$$

where $\bar{M}(s)$ has no poles or zeros in the closed right half plane.

Proof. Assume there exist $\tilde{M}(s)$ satisfying the conditions of Lemma 3. Then, from Lemma 2 it follows that there exists a real polynomial $M(s)$ satisfying (3.1)-(3.3) and (2.12). Furthermore, the restriction (3.3) also requires that $m_i(s)$ have only even powers of s, such that for some integer n, $m_i(s) = \sum_{j=0}^{n} m_{ij}s^{2j}$. This implies that

$$m_i(s) = m_{in}\prod_{k=1}^{n}(s^2 - a_k) = m_{in}\prod_{k=1}^{n}(s - b_k)(s + b_k), \tag{3.5}$$

where $b_k = \sqrt{a_k}$ and here $\sqrt{\cdot}$ denotes the square root in the right half plane.

For $s \to \infty$ (3.2) implies that for some real positive scalar c, $(-1)^n m_{in} = c^2$, which in turn implies that

$$m_i(s) = c^2\prod_{k=1}^{n}(-s + b_k)(s + b_k) = n_i(-s)n_i(s), \tag{3.6}$$

where $n_i(s)$ has no zeros in the closed right half plane. Note that if $\text{Im}(b_k) \neq 0$ there exists $j \neq k$ such that $b_j = \bar{b}_k$. It follows that $n_i(s) \triangleq c\prod_{k=1}^{n}(s + b_k)$ is a real polynomial. Now it follows from

(3.6) that $M(s) = N(-s)N(s)$ where $N(s)$ is a real polynomial matrix with no zeros in the closed right half plane. If $d(s)$ is any n^{th} order, real polynomial with no zeros in the closed right half plane and $M'(s) \triangleq \bar{M}(-s)\bar{M}(s)$ where $\bar{M}(s) \triangleq \frac{N(s)}{d(s)}$, then $\bar{M}'(s)$ has no zeros or poles on the imaginary axis, satisfies (3.1)-(3.3) and from Lemma 2 also satisfies (2.12).

Finally, note that each $M(s)$ given by (3.1) and (3.4), where $\bar{M}(s)$ has no poles or zeros in the closed right half plane, satisfies (3.2) and (3.3). $\triangle$

Remark 5. Equation (3.4) corresponds to the stable coprime factorization of $M(s)$ given by (2.15) with $M_A(s) = \bar{M}(s)$ and $M_B(s) = \bar{M}^{-1}(s)$. Furthermore, if $\bar{M}(s)$ is strictly proper then $M(s)$ given by (3.4) is strongly positive real and $\bar{M}(s)$ and $\bar{M}^{-1}(s)$ both have state space realizations.

Remark 6. $\bar{M}(s)$ is precisely an asymptotically stable, minimum phase transfer function representation of the classical D-scales from complex structured singular value analysis.[1, 2]

From Remark 5, it follows that if we denote the state space realization of strictly proper $M_A(s)$ in (2.15) by

$$M_A(s) \sim \left[\begin{array}{c|c} \bar{A} & \bar{B} \\ \hline \bar{C} & \bar{D} \end{array}\right], \tag{3.7}$$

then, we can simply choose $M_B(s) = {M_A}^{-1}(s)$ and hence using a standard state space realization inversion formula [35]

$$M_B(s) \sim \left[\begin{array}{c|c} \bar{A} - \bar{B}\bar{D}^{-1}\bar{C} & \bar{B}\bar{D}^{-1} \\ \hline -\bar{D}^{-1}\bar{C} & \bar{D}^{-1} \end{array}\right]. \tag{3.8}$$

Details on how to choose $(\bar{A}, \bar{B}, \bar{C}, \bar{D})$ to enforce the block-diagonal structure of $M_A(s)$ are given in Reference 36.

Now, if we let $T_\gamma(s)$ have the state space realization $(A_\gamma, B_\gamma, C_\gamma, D_\gamma)$ as in (2.14), then, referring to (2.16),

$$M_A(s)T_\gamma(s)M_B(s) \sim \left[\begin{array}{c|c} \tilde{A}_\gamma & \tilde{B}_\gamma \\ \hline \tilde{C}_\gamma & \tilde{D}_\gamma \end{array}\right],$$

where

$$\tilde{A}_\gamma = \begin{bmatrix} \bar{A} - \bar{B}\bar{D}^{-1}\bar{C} & 0 & 0 \\ -B_\gamma\bar{D}^{-1}\bar{C} & A_\gamma & 0 \\ -\bar{B}D_\gamma\bar{D}^{-1}\bar{C} & \bar{B}B_\gamma & \bar{A} \end{bmatrix}, \; \tilde{B}_\gamma = \begin{bmatrix} \bar{B}\bar{D}^{-1} \\ B_\gamma\bar{D}^{-1} \\ \bar{B}D_\gamma\bar{D}^{-1} \end{bmatrix}, \tag{3.9}$$

$$\tilde{C}_\gamma = \begin{bmatrix} -\bar{D}D_\gamma\bar{D}^{-1}\bar{C} & \bar{D}D_\gamma & \bar{C} \end{bmatrix}, \quad \tilde{D}_\gamma = \bar{D}D_\gamma\bar{D}^{-1}. \tag{3.10}$$

The next theorem which considers complex, block-structured uncertainty, follows immediately from Corollary 1 and Lemma 1.

Theorem 4. Let $\mathcal{I}_i = \mathbf{\Delta}^{\mathrm{I}}$, $i = 1, \ldots, p$, and suppose $G(s)$ is asymptotically stable. If there exist $(\bar{A}, \bar{B}, \bar{C}, \bar{D})$, $P > 0$, and $\epsilon > 0$ such that $\tilde{D}_\gamma + \tilde{D}_\gamma^T > 0$ and

$$0 = \tilde{A}_\gamma{}^T P + P\tilde{A}_\gamma + (\tilde{B}_\gamma{}^T P - \tilde{C}_\gamma)^T (\tilde{D}_\gamma + \tilde{D}_\gamma{}^T)^{-1} (\tilde{B}_\gamma{}^T P - \tilde{C}_\gamma) + \epsilon I,$$

where $\tilde{A}_\gamma, \tilde{B}_\gamma, \tilde{C}_\gamma, \tilde{D}_\gamma$ are given by (3.9) and (3.10), then the negative feedback interconnection of $G(s)$ and Δ is asymptotically stable for all $\Delta \in \mathbf{\Delta}_\gamma$.

3.2. Real, Diagonal Uncertainty

Recall from (2.10) and (2.19) that a multiplier corresponding to real, diagonal uncertainty (with possible repeated elements) is given by

$$M(s) = \text{block-diag}(M_1(s), \ldots, M_p(s)),$$

where

$$\text{He}[M(j\omega)] \geq 0, \ \omega \in \mathcal{R} \cup \infty. \tag{3.11}$$

Now, if we let $M(s) = M_{\mathrm{B}}{}^*(s)^{-1} M_{\mathrm{A}}(s)$ as in (2.15), then it follows from Corollary 1 that if we consider only strict inequality in (3.11), then we can replace (3.11) with

$$\text{He}[M_{\mathrm{A}}(j\omega) M_{\mathrm{B}}(j\omega)] > 0, \ \omega \in \mathcal{R} \cup \infty. \tag{3.12}$$

Next let the state space realizations of $M_{\mathrm{A}}(s)$ and $M_{\mathrm{B}}(s)$ be denoted, respectively, by

$$M_{\mathrm{A}}(s) \sim \left[\begin{array}{c|c} A_{\mathrm{A}} & B_{\mathrm{A}} \\ \hline C_{\mathrm{A}} & D_{\mathrm{A}} \end{array}\right], \quad M_{\mathrm{B}}(s) \sim \left[\begin{array}{c|c} A_{\mathrm{B}} & B_{\mathrm{B}} \\ \hline C_{\mathrm{B}} & D_{\mathrm{B}} \end{array}\right].$$

If we let $T_\gamma(s)$ have the state space realization $(A_\gamma, B_\gamma, C_\gamma, D_\gamma)$ as in (2.14), then referring to (2.16)

$$M_{\mathrm{A}}(s) T_\gamma(s) M_{\mathrm{B}}(s) \sim \left[\begin{array}{c|c} \tilde{A}_{\gamma,1} & \tilde{B}_{\gamma,1} \\ \hline \tilde{C}_{\gamma,1} & \tilde{D}_{\gamma,1} \end{array}\right],$$

where

$$\tilde{A}_{\gamma,1} = \begin{bmatrix} A_{\mathrm{B}} & 0 & 0 \\ B_\gamma C_{\mathrm{B}} & A_\gamma & 0 \\ B_{\mathrm{A}} D_\gamma C_{\mathrm{B}} & B_{\mathrm{A}} C_\gamma & A_{\mathrm{A}} \end{bmatrix}, \ \tilde{B}_{\gamma,1} = \begin{bmatrix} B_{\mathrm{B}} \\ B_\gamma D_{\mathrm{B}} \\ B_{\mathrm{A}} D_\gamma D_{\mathrm{B}} \end{bmatrix},$$

$$\tilde{C}_{\gamma,1} = \begin{bmatrix} D_{\mathrm{A}} D_\gamma D_{\mathrm{B}} & D_{\mathrm{A}} C_\gamma & C_{\mathrm{A}} \end{bmatrix}, \quad \tilde{D}_{\gamma,1} = D_{\mathrm{A}} D_\gamma D_{\mathrm{B}}.$$

Similarly, referring to (3.12),

$$M_{\mathrm{A}}(s) M_{\mathrm{B}}(s) \sim \left[\begin{array}{c|c} \tilde{A}_{\gamma,2} & \tilde{B}_{\gamma,2} \\ \hline \tilde{C}_{\gamma,2} & \tilde{D}_{\gamma,2} \end{array}\right],$$

where

$$\tilde{A}_{\gamma,2} = \begin{bmatrix} A_{\rm B} & 0 \\ B_{\rm A}C_{\rm B} & A_{\rm A} \end{bmatrix}, \quad \tilde{B}_{\gamma,2} = \begin{bmatrix} B_{\rm B} \\ B_{\rm A}D_{\rm B} \end{bmatrix},$$
$$\tilde{C}_{\gamma,2} = \begin{bmatrix} D_{\rm A}C_{\rm B} & C_{\rm A} \end{bmatrix}, \quad \tilde{D}_{\gamma,2} = D_{\rm A}D_{\rm B}.$$

Now, let

$$\tilde{A}_\gamma = \text{block-diag}\{\tilde{A}_{\gamma,1}, \tilde{A}_{\gamma,2}\}, \quad \tilde{B}_\gamma = \text{block-diag}\{\tilde{B}_{\gamma,1}, \tilde{B}_{\gamma,2}\}, \tag{3.13}$$

$$\tilde{C}_\gamma = \text{block-diag}\{\tilde{C}_{\gamma,1}, \tilde{C}_{\gamma,2}\}, \quad \tilde{D}_\gamma = \text{block-diag}\{\tilde{D}_{\gamma,1}, \tilde{D}_{\gamma,2}\}. \tag{3.14}$$

The next theorem, follows immediately from Corollary 1 and Lemma 1.

Theorem 5. Let $\mathcal{I}_i = \mathbf{\Delta}^{\rm III}$, $i = 1, ..., p$, and suppose $G(s)$ is asymptotically stable. If there exist $(A_{\rm A}, B_{\rm A}, C_{\rm A}, D_{\rm A})$, $(A_{\rm B}, B_{\rm B}, C_{\rm B}, D_{\rm B})$, $P > 0$, and $\epsilon > 0$ such that $\tilde{D}_\gamma + \tilde{D}_\gamma^T > 0$ and

$$0 = \tilde{A}_\gamma{}^T P + P\tilde{A}_\gamma + (\tilde{B}_\gamma{}^T P - \tilde{C}_\gamma)^T (\tilde{D}_\gamma + \tilde{D}_\gamma{}^T)^{-1} (\tilde{B}_\gamma{}^T P - \tilde{C}_\gamma) + \epsilon I,$$

where $\tilde{A}_\gamma, \tilde{B}_\gamma, \tilde{C}_\gamma, \tilde{D}_\gamma$ are given by (3.13) and (3.14), then the negative feedback interconnection of $G(s)$ and Δ is asymptotically stable for all $\Delta \in \mathbf{\Delta}_\gamma$.

3.3. Problem Formulation

Both Theorems 4 and 5 provide robustness tests in terms of the following feasibility problem.

Riccati Equation Feasibility Problem (REFP). Find $\theta \in \mathcal{R}^q$, $\epsilon > 0$, and $P \in \mathcal{R}^{r \times r}$ such that

$$0 = \tilde{A}_\gamma{}^T(\theta)P + P\tilde{A}_\gamma(\theta) + (\tilde{B}_\gamma{}^T(\theta)P - \tilde{C}_\gamma(\theta))^T (\tilde{D}_\gamma(\theta) + \tilde{D}_\gamma{}^T(\theta))^{-1} (\tilde{B}^T(\theta)P - \tilde{C}_\gamma(\theta)) + \epsilon I, \tag{3.15}$$

$$P > 0, \quad \tilde{D}_\gamma(\theta) + \tilde{D}_\gamma^T(\theta) > 0, \tag{3.16}$$

where the dimension q is determined by the multiplier and r is determined by both the multiplier and the nominal plant size. In Theorems 4 and 5, θ corresponds to the free parameters of the matrices providing a state-space representation of the multiplier factors $M_{\rm A}(s)$ and $M_{\rm B}(s)$. For example, considering Theorem 4, if all of the elements of the matrices $\bar{A}, \bar{B}, \bar{C}$, and $\bar{D}$ in (3.7) and (3.8) are free, then θ is defined by $\theta = \left(\text{vec}^T(\bar{A}), \text{vec}^T(\bar{B}), \text{vec}^T(\bar{C}), \text{vec}^T(\bar{D})\right)^T$.

If we are considering control design for a plant (A_p, B_p, C_p, D_p) under a feedback controller (A_c, B_c, C_c), then, assuming negative feedback, A in (2.14) is given by

$$A = \begin{bmatrix} A_p & -B_pC_c \\ B_cC_p & A_c - B_cD_pC_c \end{bmatrix}. \tag{3.17}$$

Hence, in Theorems 4 and 5 $\tilde{A}_\gamma$ is linear in the controller matrices. The controller matrices essentially provide extra degrees of freedom to satisfy the Riccati equation constraint (3.15). To illustrate, if all of the elements of the matrices $\bar{A}$, $\bar{B}$, $\bar{C}$, and $\bar{D}$ are free, and all of the matrices A_c, B_c, C_c are also free then θ is defined by $\theta = (\text{vec}^T(\bar{A}), \text{vec}^T(\bar{B}), \text{vec}^T(\bar{C}), \text{vec}^T(\bar{D}), \text{vec}^T(A_c), \text{vec}^T(B_c), \text{vec}^T(D_c))^T$. Note that $\tilde{A}_\gamma(\theta)$, $\tilde{B}_\gamma(\theta)$, $\tilde{C}_\gamma(\theta)$, and $\tilde{D}_\gamma(\theta)$ are generally nonlinear functions of θ. Hence it is not possible to convert the REFP to an LMI feasibility problem.

We approach the development of a solution technique by defining the following artificial cost function

$$J(\theta, \epsilon, P) = \alpha\theta^T\theta + \alpha\epsilon^2 + r_d \text{ tr } [\tilde{D}_\gamma(\theta) + \tilde{D}_\gamma^T(\theta)]^{-1} + r_p \text{ tr } P^{-1} + r_\epsilon \frac{1}{\epsilon}, \tag{3.18}$$

where α, r_d, r_p, and r_ϵ are positive scalars. This barrier cost function has the nice properties that

1. $J(\theta, \epsilon, P)$ becomes unbounded if one of the eigenvalues of P or $\tilde{D}_\gamma(\theta) + \tilde{D}_\gamma^T(\theta)$ approaches zero or ϵ approaches zero; and

2. the second derivative of the first two terms with respect to the vector $[\theta, \epsilon]^T$ is $2\alpha I$.

Property (1) is used to enforce the constraints $P > 0$, $\tilde{D}_\gamma(\theta) + \tilde{D}_\gamma^T(\theta) > 0$, and $\epsilon > 0$. Property (2) is used to improve the numerical conditioning of the Hessian matrix, which sometimes improves the numerical robustness of the associated numerical algorithms.

Remark 7. Replace ϵI in (3.15) with ϵR where $R = E^T E$ and E is the output matrix corresponding to the performance variable and let $V = D_w D_w^T$ where D_w is the input matrix corresponding to the plant disturbance. Then, if we choose $J(\theta, \epsilon, P) = \text{tr } PV +$ an additional nonnegative, multiplier-dependent term, then $J(\cdot)$ is an upper bound on an H_2 cost functional. This type of cost functional is the basis for the continuation and homotopy algorithms of References 14-17, 34, 37.

Let $\mathcal{S}$ denote the set of solutions (θ, ϵ, P) to the REFP and consider the optimization problem

$$\min_{\theta, \epsilon} J(\theta, \epsilon, P) \text{ subject to (3.15).} \tag{3.19}$$

We will attempt to find $(\theta, \epsilon, P) \in \mathcal{S}$ by solving the optimization problem (3.19). That is, we will search for $(\theta, \epsilon, P) \in \mathcal{S}$ that is an extremal to (3.19).

Note that we are assuming that if $\mathcal{S}$ is nonempty, (3.19) has a solution. However, this has not been proven. A sufficient condition to guarantee a solution is that the feasible level set

$$\Pi(\beta) = \{(\theta, \epsilon, P) \in \mathcal{S} : J(\theta, \epsilon, P) \leq \beta\},$$

is compact for $\beta \geq 0$. Clearly, $\Pi(\beta)$ is bounded due to the terms $\alpha\theta^T\theta + \alpha\epsilon^2$ and the positivity of each of the terms in (3.18). However, whether $\Pi(\beta)$ is closed depends on the closure of the set $\mathcal{S}$. The characterization of $\mathcal{S}$ is a subject of future research.

To characterize the extremals, we define the Lagrangian

$$\mathcal{L}(\theta, \epsilon, P, Q) = J(\theta, \epsilon, P) + \text{tr}\, QW(\theta, \epsilon, P), \tag{3.20}$$

where $W(\theta, \epsilon, P)$ denotes the right hand side of (3.15) and Q is the symmetric Lagrange multiplier. Note that the constraints (3.16) are absorbed into J as barriers. The necessary conditions for a solution to (3.19) are given by[25]

$$0 = \frac{\partial \mathcal{L}}{\partial \theta}, \quad 0 = \frac{\partial \mathcal{L}}{\partial \epsilon}, \tag{3.21}$$

$$0 = \frac{\partial \mathcal{L}}{\partial Q} = \tilde{A}_\gamma{}^T(\theta)P + P\tilde{A}_\gamma(\theta) + (\tilde{B}_\gamma{}^T(\theta)P - \tilde{C}_\gamma(\theta))^T(\tilde{D}_\gamma(\theta) + \tilde{D}_\gamma{}^T(\theta))^{-1}(\tilde{B}^T(\theta)P - \tilde{C}_\gamma(\theta)) + \epsilon I, \tag{3.22}$$

$$0 = \frac{\partial \mathcal{L}}{\partial P} = \tilde{F}_\gamma Q + Q\tilde{F}_\gamma^T - r_p(P^{-2}) + \frac{\beta}{\epsilon}V^T, \tag{3.23}$$

where

$$\tilde{F}_\gamma = \tilde{A}_\gamma - \tilde{B}_\gamma\left[\tilde{D}_\gamma + \tilde{D}_\gamma^T\right]^{-1}\left[\tilde{B}_\gamma P - \tilde{C}_\gamma\right].$$

Although (3.21)-(3.23) characterizes extremals to (3.19), we have not yet developed a reliable method to compute these extremals. Note that standard interior point descent methods (e.g., steepest descent, conjugate gradient, or quasi-Newton methods) cannot be directly applied due to the nature of the constraints. For example, suppose we attempt to initialize one of these methods with a multiplier (in the class of multipliers for the given uncertainty set) represented by θ_0 and also choose an initial ϵ denoted by ϵ_0. Then, if there exists a positive-definite solution P_0 to (3.15), the REFP is solved and there is no need for further computations. However, suppose there is no positive definite solution P_0 to (3.15). Then, $(\theta_0, \epsilon_0, P_0)$ cannot be used to initialize an interior point descent method to find a solution to the optimization problem (3.19) since this class of methods requires an initial feasible interior point. What is needed is a numerical technique that is able to find a solution to (3.19) by starting with a nonfeasible point $(\theta_0, \epsilon_0, P_0)$. This is accomplished in the next section using a probabililty-one homotopy algorithm.

4. Probability-One Homotopy Algorithms for Robust Controller Synthesis

Homotopies have traditionally been studied as a part of topology and have found significant application in nonlinear functional analysis and differential geometry. However, it has only been in the last two decades that homotopies have been used for practical numerical computation. The

algorithms described here are probability one, globally convergent homotopy algorithms.[26, 27] This class of algorithms is related to, but distinct from, other homotopy methods such as continuation methods.[28] An advantage of probability-one algorithms is that under certain conditions, it is possible to guarantee convergence of the algorithm from an arbitrary starting point with probability one. (This does not imply that the algorithms are guaranteed to converge to global optimum, only that they can converge to a local extremum from arbitrary starting points.) These algorithms also relieve some of the technical difficulties sometimes encountered when implementing alternative homotopy methods. For example, they allow the zero curve of the homotopy map to be nonmonotonic in the homotopy parameter λ, and avoid singularities along the zero curve.

Consider a function $F : \mathcal{R}^N \times \mathcal{R} \to \mathcal{R}^N$ that is $\mathcal{C}^2$. Given $\gamma_f \in \mathcal{R}$, it is desired to find $x \in \mathcal{R}^N$ such that

$$0 = F(x, \gamma_f). \tag{4.1}$$

This is a standard zero finding problem. In the context of the robustness analysis results of the previous section

$$x = (\theta, \epsilon), \quad N = q + 1, \tag{4.2}$$

$$F(x, \gamma) = \left(\frac{\partial \mathcal{L}}{\partial \theta}, \frac{\partial \mathcal{L}}{\partial \epsilon} \right), \tag{4.3}$$

and γ_f corresponds to some desired lower bound on the multivariable stability margin. Note that $0 = \frac{\partial \mathcal{L}}{\partial Q}$ and $0 = \frac{\partial \mathcal{L}}{\partial P}$ are implicitly satisfied by choosing P as the (maximal) solution of the Riccati equation (3.22) (or (3.15)) and Q as the solution of the Lyapunov equation (3.23).

Let $x_0 = [\theta_0, \epsilon_0]$ represent a feasible multiplier, a stablizing compensator and a positive ϵ. Furthermore let γ_0 be chosen small enough such that there exists a positive-definite solution P_0 to (3.15). (It is assumed that $\gamma_0 < \gamma_f$ such that the robustness problem is not trivial.)

We let

$$\gamma(\lambda) = (1 - \lambda)\gamma_0 + \lambda\gamma_f \tag{4.4}$$

and define the probability-one homotopy map $\rho : [0, 1) \times \mathcal{R}^N \to \mathcal{R}^N$ by

$$\rho(\lambda, x) = \lambda F(x, \gamma(\lambda)) + (1 - \lambda)(x - x_0). \tag{4.5}$$

Obviously, $0 = \rho$ has the unique solution x_0 and $\rho = F(x, \gamma_f)$. These are necessary conditions for the homotopy map. In the context of the robustness problem of Section 3, this homotopy map has the desirable property that it can be initialized with any feasible multiplier. In addition, for any $\lambda \in [0, 1)$ the corresponding point on the zero curve (x, λ) corresponds to a controller and multiplier that guarantees the level of robustness corresponding to $\gamma(\lambda)$ since the Riccati equation (3.15) (or

(3.22)) with the constraints (3.16) are satisfied with $\gamma = \gamma(\lambda)$. Hence, each point on the zero curve $(0 = \rho(\lambda, x),\ \lambda \in [0, 1))$, is physically meaningful even though $F(x, \gamma(\lambda)) \neq 0$ for $0 < \lambda < 1$.

4.1. Probability-One Homotopy Algorithm

Complete details of the numerical algorithm are in Reference 27. A sketch follows.

1. Set $\lambda \triangleq 0,\ x \triangleq x_0$.
2. Evaluate the homotopy map ρ and the Jacobian of the homotopy map $D\rho$.
3. Predict the next point $Z^{(0)}$ on the homotopy zero curve using, e.g., a Hermite cubic interpolant.
4. For $k = 0, 1, 2, ...$ until convergence do

$$Z^{(k+1)} = Z^{(k)} - [D\rho(Z^{(k)})]^{\dagger}\rho(Z^{(k)}),$$

   where $[D\rho(Z)]^{\dagger}$ is the Moore-Penrose pseudoinverse of $D\rho(Z)$. Let $(x_1, \lambda_1) = \lim_{k\to\infty} Z^k$.
5. If $\lambda_1 < 1$, then set $x = x_1,\ \lambda = \lambda_1$, and go to to step (2).
6. If $\lambda_1 > 1$, compute the solution x at $\lambda = 1$ using, e.g., inverse linear interpolation.[27]

5. The Popov Multiplier and an Illustrative Numerical Example

In this section we consider the special case of the Popov multiplier. In particular we let $M(s) = H^2 + Ns$, where H and N are real and diagonal. This multiplier is in the class of multipliers for real parametric uncertainty and can also be applied to complex uncertainty by choosing $N = 0$, in which case H is the constant D-scale matrix considered in Reference 17, 38. Note that if $M(s)$ is the Popov multiplier and if $T_\gamma(s)$ is asymptotically stable, then $M(s)T_\gamma(s)$ is asymptotically stable. Hence, the framework of Section 2 applies with $M_{\rm A}(s) = M(s)$ and $M_{\rm B}(s) = I$. Next we use the Popov multiplier to consider the case of nonrepeated, real parametric, scalar uncertainty (i.e., $\mathcal{I}_i = \mathbf{\Delta}^{\rm III}$ and $k_i = 1$ for $i = 1, ..., p$).

5.1. Nonrepeated, Real Parametric, Diagonal Uncertainty

It is important to note that for real parametric uncertainty, D, the direct feedthrough term in the state space realization of $G(s)$ in Figure 1, is always zero. Hence, the state space realization of

$T_\gamma(s)$ (see (2.13) and (2.14)) becomes

$$T_\gamma(s) \sim \left[\begin{array}{c|c} A+\gamma BC & \sqrt{2\gamma}B \\ \hline \sqrt{2\gamma}C & I \end{array}\right], \tag{5.1}$$

or, equivalently,

$$T_\gamma(s) = T_{\rm sp}(s) + I, \tag{5.2}$$

where $T_{\rm sp}(s)$ is the strictly proper transfer function given by

$$T_{\rm sp}(s) \sim \left[\begin{array}{c|c} A+\gamma BC & \sqrt{2\gamma}B \\ \hline \sqrt{2\gamma}C & 0 \end{array}\right]. \tag{5.3}$$

Note that for nonrepeated, real parametric, diagonal uncertainty the Popov multiplier (with H and N diagonal) is a compatible multiplier. In this case the robustness condition (2.12) becomes

$$\mathrm{He}[(H^2 + j\omega N)(T_{\rm sp}(j\omega) + I)] > 0, \quad \omega \in \mathcal{R} \cup \infty, \tag{5.4}$$

or, equivalently,

$$\mathrm{He}[(H^2 + j\omega N)T_{\rm sp}(j\omega)] + \mathrm{He}[H^2 + j\omega N] > 0, \quad \omega \in \mathcal{R} \cup \infty. \tag{5.5}$$

It follows from Lemma 3 of Reference 19 that

$$(H^2 + Ns)T_{\rm sp}(s) \sim \left[\begin{array}{c|c} A+\gamma BC & \sqrt{2\gamma}B \\ \hline \sqrt{2\gamma}(H^2C + NC(A+\gamma BC)) & 2\gamma NCB \end{array}\right]. \tag{5.6}$$

Then, since $\mathrm{He}[H^2 + j\omega N] = H^2 = \mathrm{He}\, H^2$, the robustness condition (5.4) is equivalent to

$$\mathrm{He}\, T_{\rm H}(j\omega) > 0, \quad \omega \in \mathcal{R} \cup \infty, \tag{5.7}$$

where

$$T_{\rm H}(s) \sim \left[\begin{array}{c|c} A+\gamma BC & \sqrt{2\gamma}B \\ \hline \sqrt{2\gamma}(H^2C + NC(A+\gamma BC)) & H^2 + 2\gamma NCB \end{array}\right]. \tag{5.8}$$

The next theorem follows immediately from (5.7), (5.8), and Lemma 1.

Theorem 6. Let $\mathcal{I}_i = \mathbf{\Delta}^{\rm IV}$, $i = 1, ..., p$, assume $G(s)$ is asymptotically stable, and define $A_\gamma \triangleq A + \gamma BC$.

1. If there exist real diagonal H and N, $P > 0$ and $\epsilon > 0$ such that $NCB + B^TC^TN + \gamma^{-1}H^2 > 0$ and

$$\begin{aligned} 0 \;=\; & A_\gamma{}^T P + PA_\gamma + \epsilon I \\ & + [B^TP - H^2C - NCA_\gamma]^T[NCB + B^TC^TN + \gamma^{-1}H^2]^{-1}[B^TP - H^2C - NCA_\gamma], \end{aligned} \tag{5.9}$$

or,

2. if there exist positive definite, diagonal $S(\triangleq H^2)$ and N, P, and $\epsilon > 0$ such that

$$\begin{bmatrix} -A_\gamma{}^T P - P A_\gamma & -\sqrt{2\gamma} PB + \sqrt{2\gamma} C^T S + \sqrt{2\gamma} A_\gamma{}^T C^T N \\ -\sqrt{2\gamma} B^T P + \sqrt{2\gamma} SC + \sqrt{2\gamma} NCA_\gamma & 2\gamma NCB + 2\gamma N B^T C^T + 2S - 2\epsilon I \end{bmatrix} \geq 0, \tag{5.10}$$

then the negative feedback interconnection of $G(s)$ and Δ is asymptotically stable for all $\Delta \in \mathbf{\Delta}_\gamma$.

5.2. Robust Control Synthesis Using the Popov Multiplier for a Benchmark Problem

To illustrate robust control synthesis with the probability-one homotopy algorithm, we consider the two-mass/spring benchmark system shown in Figure 2 with uncertain stiffness k. A control force acts on the body 1 and the position of body 2 is measured, resulting in a noncolocated control problem. This benchmark problem is discussed in detail in Reference 39.

The open-loop plant (for $m_1 = m_2 = 1$) is given by

$$\dot{x}_p(t) = A_p(k) x_p(t) + B_p u(t) + D_1 w(t), \tag{5.11}$$

$$y(t) = C_p x_p(t) + D_2 w(t), \tag{5.12}$$

$$z(t) = E_1 x_p(t), \tag{5.13}$$

where $z = x_2$ is the performance variable, y is a noise corrupted measurement of x_2,

$$x_p^T = [x_1, x_2, x_3, x_4], \quad A_p(k) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k & k & 0 & 0 \\ k & -k & 0 & 0 \end{bmatrix}, \quad B_p = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

$$D_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad C_p = E_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}, \quad D_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

We desire to design a constant gain linear feedback compensator of the form

$$\dot{x}_c(t) = A_c x_c(t) + B_c y(t), \tag{5.14}$$

$$u(t) = -C_c x_c(t), \tag{5.15}$$

such that the closed-loop system is stable for $0.5 < k < 2.0$ and for a unit impulse disturbance at $t = 0$, the performance variable z has a settling time of about 15 s for the nominal system (with $\overline{k} = k_{\text{nom}} = 1$).

We begin by constructing the uncertainty feedback system as shown in Figure 1. It is assumed that $k = k_{\text{nom}} + \Delta k$. The perturbation in $A_p(k)$ due to a change Δk in the stiffness element k from the nominal value k_{nom} is given by

$$A_p(k) - A_p(k_{\text{nom}}) \triangleq \Delta A_p = -B_o \Delta k C_o, \tag{5.16}$$

where $B_o = \begin{bmatrix} 0 & 0 & 1 & -1 \end{bmatrix}^T$, and $C_o = \begin{bmatrix} 1 & -1 & 0 & 0 \end{bmatrix}$. Assuming negative feedback, the closed-loop state matrix is given by

$$A(k) = \begin{bmatrix} A_p(k) & -B_p C_c \\ B_c C_p & A_c \end{bmatrix}. \tag{5.17}$$

Next, define $B = \begin{bmatrix} B_o & 0_{4\times 1} \end{bmatrix}$, and $C = \begin{bmatrix} C_o & 0_{1\times 4} \end{bmatrix}$ so that $G(s)$ and Δ in Figure 1 are given by $G(s) \sim \left[\begin{array}{c|c} A(k_{\text{nom}}) & B \\ \hline C & 0 \end{array}\right]$ and $\Delta = \Delta k$.

The upper bound on the H_2 cost functional discussed in Remark 7 is given by

$$J = \frac{\beta}{\epsilon} \text{tr}[(P + 2\gamma C^T N C)V], \tag{5.18}$$

where V is given by

$$V = \begin{bmatrix} V_1 & V_{12} B_c^T \\ B_c V_{12}^T & B_c V_2 B_c^T \end{bmatrix}, \tag{5.19}$$

with $V_1 = D_1 D_1^T$, $V_2 = \rho D_2 D_2^T$, and $V_{12} = \sqrt{\rho}\, D_1 D_2^T$. Also, as in Remark 7, the Riccati equation in (5.9) is modified slightly in that ϵI is replaced by ϵR where R is given by

$$R = \begin{bmatrix} R_1 & -R_{12} C_c \\ -C_c^T R_{12}^T & C_c^T R_2 C_c \end{bmatrix}, \tag{5.20}$$

where $R_1 = E_1^T E_1$, $R_2 = \rho$, and $R_{12} = \sqrt{\rho}\, E_1^T$. Here the parameter ρ is used as a design parameter to increase or decrease the authority of the controller.

It can be seen that the diagonal H and N of the Popov multiplier, reduce to scalars for this particular example. The variable x in (4.2) is given by

$$x = \begin{bmatrix} H & N & \epsilon & \text{vec}^T(A_c) & \text{vec}^T(B_c) & \text{vec}^T(C_c) \end{bmatrix}^T, \tag{5.21}$$

and consequently, the function F in (4.3), is given by

$$F(x,\gamma) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial H} & \frac{\partial \mathcal{L}}{\partial N} & \frac{\partial \mathcal{L}}{\partial \epsilon} & \text{vec}^T\left(\frac{\partial \mathcal{L}}{\partial A_c}\right) & \text{vec}^T\left(\frac{\partial \mathcal{L}}{\partial B_c}\right) & \text{vec}^T\left(\frac{\partial \mathcal{L}}{\partial C_c}\right) \end{bmatrix}^T. \tag{5.22}$$

The initial point x_0 is chosen in the following manner. H_0, N_0, and ϵ_0, are chosen arbitrarily as 10, 10, and 1, respectively. The initial controller $(A_{c,0}, B_{c,0}, C_{c,0})$ is an LQG controller for the

nominal plant corresponding to $\rho = 0.001$. No robustness is expected of this controller and hence γ_0 in (4.4) is chosen close to zero (i.e., $\gamma_0 = 0.01$).

The controller transfer function obtained by the probability-one homotopy algorithm in Subsection 4.1 is given by

$$H(s) = \frac{2819\,(s + 0.2079)[(s - 0.0978)^2 + 0.8063^2]}{[(s + 4.004)^2 + 1.8294^2][(s + 3.4747)^2 + 9.9745^2]}. \tag{5.23}$$

This controller is guaranteed by the theory to be robust for the range $0.5 < k < 2.0$ and this was also verified by a direct search. The settling time for the system was chosen to be the time required for the displacement of mass 2 to reach and stay within the interval [-0.1m, 0.1m]. The controller is seen to satisfy the settling time objective when connected to the nominal model corresponding to $k = 1$ N/m, as can be seen from the impulse response of the closed-loop system in Figure 3.

6. Conclusion

This paper has demonstrated that fixed-architecture, robust control design using general fixed-structure multipliers can be formulated as a Riccati equation feasibility problem, a nonlinear, algebraic feasibility problem. Probability-one homotopy algorithms were proposed to solve this feasibility problem. These algorithms differ from previously developed continuation algorithms, developed exclusively for the case of the Popov multiplier, in that they can be initialized with any admissible multiplier and stabilizing compensator. Like other probability-one homotopy algorithms they also tend to be better conditioned than the alternative continuation algorithms. The results were specialized to the special case of Popov multipliers and fixed-architecture robust control design was illustrated using a standard benchmark problem.

The key step in the practical implementation of these homotopy algorithms is the numerical implementation of the computation of the Jacobian of the homotopy map. Analytical expressions for the Jacobian tend to be complex and difficult to derive due to this complexity. For the practical implementation of homotopy algorithms for more general forms of the multipliers it is important to develop reliable symbolic matrix differentiation routines. The special matrix structures that arise in the computation of the Jacobian must also be exploited to speed up the algorithms. An illustration of this is given in Reference 40.

References

1. Doyle. J. C., 'Analysis of Feedback Systems with Structured Uncertainties', *Proceedings of the IEE - Part D*, pp. 242-250, 1982.

2. Packard. A. and J. C. Doyle, 'The Complex Structured Singular Value', *Automatica*, **29**, 71-109 (1993).

3. Safonov. M. G., *Stability and Robustness of Multivariable Feedback Systems*, MIT Press, Cambridge, MA. 1980.

4. Safonov. M. G. and M. Athans, 'A Multiloop Generalization of the Circle Criterion for Stability Margin Analysis', *IEEE Transactions on Automatic Control*, **AC-26**, 415-422 (1981).

5. Fan, M. K. H., A. L. Tits, and J. C. Doyle, 'Robustness in the Presence of Mixed Parametric Uncertainty and Unmodeled Dynamics', *IEEE Transactions on Automatic Control*, **AC-36**, 25-38 (1991).

6. Young. P. M., *Robustness with Parametric and Dynamic Uncertainty*, Ph.D. Dissertation, California Institute of Technology, Pasadena, CA, 1993.

7. Haddad. W. M. and D. S. Bernstein, 'Parameter-Dependent Lyapunov Functions, Constant Real Parameter Uncertainty, and the Popov Criterion in Robust Analysis and Synthesis', Part 1, Part 2. *Proceedings of the IEEE Conference on Decision and Control*, Brighton, U.K. , pp. 2274-2279, pp. 2632-2633, Dec. 1991.

8. Haddad. W. M., J. P. How, S. R. Hall, and D. S. Bernstein, 'Extensions of Mixed-μ Bounds to Monotonic and Odd Monotonic Nonlinearities Using Absolute Stability Theory', Part 1, Part 2, *Proceedings of the IEEE Conference on Decision and Control*, Tuscon, AZ, pp. 2813-2819, pp. 2820-2823, Dec. 1992.

9. Haddad. W. M. and D. S. Bernstein, 'Explicit Construction of Quadratic Lyapunov Functions for the Small Gain, Positivity , Circle, and Popov Theorems and their Application to Robust Stability Part I: Continuous-Time Theory', *International Journal of Robust and Nonlinear Control*. **3**, 313-339 (1993).

10. Haddad. W. M., J. P. How, S. R. Hall, and D. S. Bernstein, 'Extensions of Mixed-μ Bounds to Monotonic and Odd Monotonic Nonlinearities Using Absolute Stability Theory', *International Journal of Control*, **60**, 905-951 (1994).

11. Haddad. W. M. and D. S. Bernstein, 'Parameter-Dependent Lyapunov Functions and the Popov Criterion in Robust Analysis and Synthesis', *IEEE Transactions on Automatic Control*, **AC-40**. 536-543 (1995).

12. Haddad. W. M., D. S. Bernstein, and V. S. Chellaboina, 'Generalized Mixed-μ Bounds for Real and Complex Multiple-Block Uncertainty with Internal Matrix Structure', *Proceedings of the American Control Conference*, Seattle, WA, pp. 2843-2847, June 1995.

13. How, J. P. and S. R. Hall, 'Connections Between the Popov Stability Criterion and Bounds for Real Parameter Uncertainty', *Proceedings of the American Control Conference*, San Francisco, CA, pp. 1084-1089, June 1993.

14. How, J. P., W. M. Haddad, and S. R. Hall, 'Application of Popov Controller Synthesis to Benchmark Problems with Real Parameter Uncertainty', *AIAA Journal of Guidance, Control and Dynamics*, **17**, 759-768 (1994).

15. How, J. P., S. R. Hall, and W. M. Haddad, 'Robust Controllers for the Middeck Active Control Experiment Using Popov Controller Synthesis', *IEEE Transactions on Control Systems Technology*, **2**, 73-87 (1994).

16. How, J., E. G. Collins, Jr., and W. M. Haddad, 'Optimal Popov Controller Analysis and Synthesis for Systems With Real Parameter Uncertainty', *IEEE Transactions on Control Systems Technology*, **4**, 200-207, 1996.

17. Haddad, W. M., E. G. Collins, Jr., and R. Moser, 'Structured Singular Value Controller Synthesis Using Constant D-Scales Without D-K Iteration', *Proceedings of the American Control Conference*, Baltimore, MD, pp. 2819-2823, June 1994. (Also to appear in the *International Journal of Control.*)

18. Chiang, R. Y. and M. G. Safonov, 'Real K_m-Synthesis via Generalized Popov Multipliers', *Proceedings of the American Control Conference*, Chicago, IL, pp. 2417-2418, June 1992.

19. Ly, J. H., M. G. Safonov, and R. Y. Chiang, 'Real/Complex Multivariable Stability Margin Computation via Generalized Popov Multiplier - LMI Approach', *Proceedings of the American Control Conference*, Baltimore, MD, pp. 425-429, June 1994.

20. Safonov, M. G. and R. Y. Chiang, 'Real/Complex μ Without Curve Fitting', *Control and Dynamic Systems*, **56**, New York: Academic Press, 303-324 (1993).

21. Balakrishnan, V., Y. Huang, A. Packard, and J. Doyle, 'Linear Matrix Inequalities in Analysis with Multipliers', *Proceedings of the American Control Conference*, Baltimore, MD, pp. 1228-1232, June 1994.

22. K.C. Goh, J.H. Ly, L. Turan, and M.G. Safonov, 'μ/K_m Synthesis via Bilinear Matrix Inequalities', *Proc. IEEE Conf. Dec. Contr.*, Orlando, FL, pp. 2032-2037, Dec. 1994.

23. K.C. Goh, M.G. Safonov, and G.P. Papavassilopoulus, 'Global Optimization Approaches for the Biaffine Matrix Inequality Problem', *Proc. IEEE Conf. Dec. Contr.*, Orlando, FL, pp. 2009-2014, Dec. 1994.

24. Safonov, M. G., K. C. Goh, and J. H. Ly, 'Control System Synthesis via Bilinear Matrix Inequalities', *Proceedings of the American Control Conference*, Baltimore, MD, pp. 45-49, June 1994.

25. Fletcher, R., *Practical Methods of Optimization: Second Edition*, New York.:Wiley, 1987.

26. Watson, L. T., 'Numerical Linear Algebra Aspects of Globally Convergent Homotopy Methods', *SIAM Review*, **28**, 529-545 (1987).

27. Watson, L. T., S. C. Billups, and A. P. Morgan, 'Algorithm 652: HOMPACK: A Suite of Codes for Globally Convergent Homotopy Algorithms', *ACM Transactions on Mathematical Software*, **13**, 281-310 (1987).

28. Allgower, E. L. and K. Georg, *Numerical Continuation Methods*, New York: Springer-Verlag, 1990.

29. Anderson, B.D.O and S. Vongpanitlerd, *Network Analysis and Synthesis: A Modern Systems Theory Approach*, Eaglewood Cliffs, NJ: Prentice Hall, 1973.

30. J. T. Wen, 'Time Domain and Frequency Domain Conditions for Strict Positive Realness', *IEEE Trans. Autom. Contr.*, **3**, 988-992 (1988).

31. M. Vidyasagar, *Control System Synthesis: A Factorization Approach*, Cambridge, MA: MIT Press, 1985.

32. C. N. Nett, C.A. Jacobson, and M.J. Balas. 'A Connection Between State-Space and Doubly Coprime Fractional Representations', *IEEE Trans. Autom. Contr.*, **AC-29**, 831-832 (1984).

33. M. Vidyasagar, *Nonlinear System Analysis*, New Jersey: Prentice Hall, 1993.

34. Collins, E. G. Jr., W. M. Haddad, and L. D. Davis, 'Riccati Equation Approaches for Small Gain, Positivity, and Popov Robustness Analysis', *Journal of Guidance, Control and Dynamics*. **17**, 322-329 (1994).

35. J. M. Mackiejowski, *Multivariable Feedback Design*, New York.: Addison-Wesley, 1989.

36. Haddad. W. M., E. G. Collins, Jr., and R. Moser, 'Complex Structured Singular Value Analysis Using Fixed Structure Dynamic D-Scales', *Proceedings of the IEEE Conference on Decision and Control*, Orlando, FL, pp. 3003-3008, Dec 1994. (Also to appear in *Mathematical Modeling of Systems*)

37. Ge. Y., E. G. Collins, Jr., and L. T. Watson, 'A Homotopy Algorithm for Full and Reduced Order Mixed Norm H_2/H_∞ Synthesis', *Proceedings of the IEEE Conference on Decision and Control*. Orlando, FL, pp. 2672-2677, Dec. 1994. (Also submitted to *Optimal Control Applications and Methods.*)

38. Apkarian, P., J. P. Chretien, P. Gahinet, and J. M. Biannic, 'μ Synthesis by $D - K$ Iteration with Constant Scaling', *Proceedings of the American Control Conference*, San Francisco, CA, pp. 3192-3190, June 1993.

39. Wei, B. and D. S. Bernstein, 'Benchmark Problems for Robust Control Design', *Journal of Guidance, Control, and Dynamics*, **15**, 1057-1059 (1992).

40. Collins, E. G. Jr., L. D. Davis, and S. Richter, 'Design of Reduced-Order, H_2 Optimal Controllers using a Homotopy Algorithm', *International Journal of Control*, **61**, 97-126 (1995).

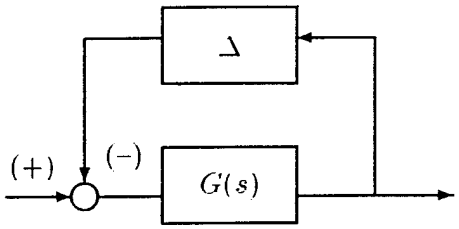

Figure 1: Standard Uncertainty Feedback Configuration

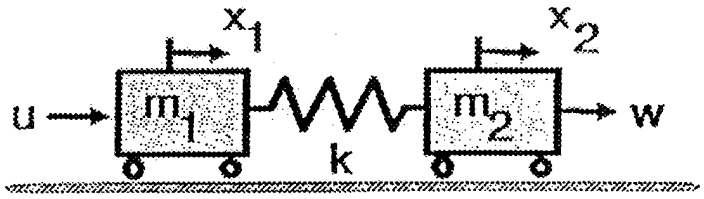

Figure 2: Two Mass/Spring System

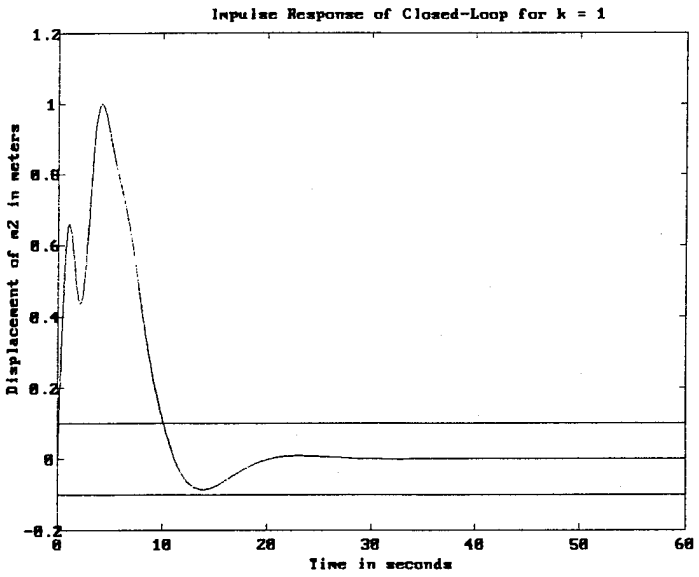

Figure 3: Impulse Response of Closed-Loop System

Figure Captions

1. Standard Uncertainty Feedback Configuration
2. Two Mass/Spring System
3. Impulse Response of Closed-Loop System

Appendix B

"A Comparison of Descent and Continuation Algorithms for H_2 Optimal Reduced-Order Control Design"

March 4, 1996

A Comparison of Descent and Continuation Algorithms for H_2 Optimal, Reduced-Order Control Design

by

Emmanuel G. Collins, Jr. and Debashis Sadhukhan
Department of Mechanical Engineering
Florida A&M University - Florida State University
2525 Pottsdamer St.
Tallahassee, FL 32310-6046
(904) 487-6373
FAX: (904) 487-6337
ecollins@eng.fsu.edu

Abstract

Reduced-order control is important in control engineering practice due to inevitable limitations on the throughput of the control processors. This paper considers the direct design of H_2 optimal, reduced-order controllers via parameter optimization approaches. Two classes of numerical methods are considered: 1) descent methods, and 2) continuation (or homotopy) methods. In particular, four standard descent methods are compared with a recently developed continuation algorithm by using three examples appearing in the literature. The continuation algorithm is seen to be much more reliable than the descent methods.

1. Introduction

One of the deficiencies of modern control laws, developed by simply solving a pair of decoupled Riccati equations, in particular, linear-quadratic-guassian (LQG) control and full-order H_∞ control, is that the resultant control laws are always of the order of the design plant. These techniques, though relatively easy to implement computationally, do not allow the designer to constrain the architecture (e.g. order and degree of centralization) of the controller. Such constraints are often necessary in engineering practice due to throughput limitations of the control processors. Reduced-order control is therefore of paramount importance in practical control design. This paper focuses on the design of H_2 optimal, reduced-order controllers.

Two main approaches have been developed to solve the H_2 optimal, reduced-order design problem. The first approach attempts to develop approximations to the optimal reduced-order controller

by reducing the dimension of an LQG controller(Yousuff and Skelton 1984a, Yousuff and Skelton 1984b, Anderson and Liu 1989, Villemagne and Skelton 1988, Liu *et al. 1990*). These methods are attractive because they require relatively little computation and should be used if possible. Unfortunately, they tend to yield controllers that either destabilize the system or have poor performance as the requested controller dimension is decreased or the requested control authority level is increased. Hence, if used in isolation, these methods do not yield a reliable methodology for reduced-order design. In addition, these methods do not extend to the design of decentralized controllers. However, it should be mentioned that, in regards to reduced-order control design, the indirect approaches at worst are valuable in providing good initial conditions for the direct approaches described below.

In contrast to controller reduction, direct approaches attempt to directly synthesize an optimal, reduced-order (or decentralized) controller by a numerical optimization scheme. There are two main classes of parameter optimization approaches to direct control design. The first class relies on the use of descent methods (Kramer and Calise 1987, Kuhn and Schmidt 1987, Kwakernaak and Sivan 1972, Ly *et al. 1985*, Mukhopadhyay 1982, Mukhopadhyay 1987, Voth and Ly 1991). Algorithms in this class reduce the H_2 cost at each iteration. The second class relies on the use of continuation methods (Collins *et al.* 1994, Mercadal 1991). In contrast to the descent methods, the H_2 cost is not necessarily reduced at each iteration. It should be mentioned that continuation algorithms (Collins *et al.* 1996b) have also been developed to solve the "optimal projection equations," a set of four coupled Lyapunov and Riccati equations that characterize the H_2 optimal, reduced-order compensator. However, this approach will not be considered here.

From a practical design perspective it is important to determine which class of methods tends to be more numerically robust. As with the vast majority of numerical methods for nonconvex optimization problems, answers to these questions are extremely difficult to prove analytically. Instead, we must rely on numerical experimentation to observe trends. Hence, in this paper the behavior of some standard descent methods (i.e., steepest descent, conjugate gradient, BFGS Quasi-Newton, and Newton's method) (Fletcher 1987) are compared to the corresponding behavior of the continuation algorithm of (Collins *et al.* 1994) by considering design for three reduced-order control design problems appearing in the literature. The results clearly indicate that the continuation algorithm tends to be more numerically robust and is most efficient when the controller is constrained to a tridiagonal form.

The paper is organized as follows. Section 2 formulates the H_2 optimal, reduced-order dynamic compensation problem as a constrained parameter optimization problem and discusses various basis options for the controller in order to reduce the size of the controller parameter vector. Section

3 briefly describes the algorithms for the descent and continuation methods for the H_2 optimal, reduced-order control problem. The descriptions emphasize how the constraint that the controller be stabilizing is taken into account in the algorithms. Section 4 uses the three examples mentioned above to present a comparison of both the numerical robustness and speed of convergence of the descent and continuation methods. Finally, Section 5 presents conclusions.

Notation

$\mathcal{R}^n$	$n \times 1$ real vector
z^T	transpose of z
$\mathrm{tr}(M)$	trace of the square matrix M
E	expectation operator
$\mathrm{dom}(g)$	domain of the function $g(\cdot)$

2. H_2 Optimal, Reduced-Order Dynamic Compensation

2.1. Problem Formulation

Consider the system

$$\dot{x}(t) = Ax(t) + Bu(t) + D_1 w(t) \tag{2.1}$$

$$y(t) = Cx(t) + Du(t) + D_2 w(t) \tag{2.2}$$

$$z(t) = E_1 x(t) + E_2 u(t) \tag{2.3}$$

where $x \in \mathcal{R}^{n_x}$, $u \in \mathcal{R}^{n_u}$, $y \in \mathcal{R}^{n_y}$, $z \in \mathcal{R}^{n_z}$, $w \in \mathcal{R}^{n_w}$ is white noise with unit intensity, D_2 has full row rank, and E_2 has full column rank. We desire to design a n_c^{th} order dynamic compensator,

$$\dot{x}_c(t) = A_c x_c(t) + B_c y(t) \tag{2.4}$$

$$u(t) = -C_c x_c(t) \tag{2.5}$$

where $n_c \leq n$, which minimizes the steady state performance criterion

$$J(A_c, B_c, C_c) \triangleq \lim_{t \to \infty} E[z^T(t) z(t)]. \tag{2.6}$$

In the frequency domain, the cost in (2.6) may be interpreted as

$$J(A_c, B_c, C_c) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \mathrm{tr}[G_{zw}(j\omega) G_{zw}^*(j\omega)] d\omega \tag{2.7}$$

where $G_{zw}(s)$ is the transfer function from w to z and the right hand side is the square of the H_2 norm of $G_{zw}(s)$.

The state-space evolution of the closed-loop system corresponding to (2.1)-(2.5) is described by

$$\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{D}w(t) \tag{2.8}$$

where

$$\tilde{x}(t) \triangleq \begin{bmatrix} x(t) \\ x_c(t) \end{bmatrix}, \quad \tilde{A} \triangleq \begin{bmatrix} A & -BC_c \\ B_cC & A_c - B_cDC_c \end{bmatrix}, \quad \tilde{D} \triangleq \begin{bmatrix} D_1 \\ B_cD_2 \end{bmatrix}. \tag{2.9}$$

Now, the cost (2.6) can be expressed as

$$J(A_c, B_c, C_c) = \lim_{t\to\infty} E[\tilde{x}^T(t)\tilde{R}\tilde{x}(t)] \tag{2.10}$$

where

$$\tilde{R} \triangleq \begin{bmatrix} R_1 & R_{12}C_c \\ C_c^T R_{12}^T & C_c^T R_2 C_c \end{bmatrix}, \quad R_1 \triangleq {E_1}^T E_1, \quad R_{12} \triangleq 2{E_1}^T E_2, \quad R_2 \triangleq {E_2}^T E_2. \tag{2.11}$$

(Note that since E_2 has full column rank, $R_2 > 0$.)

To guarantee that the cost J is finite and independent of initial conditions we restrict our attention to the set of stabilizing compensators, $\mathcal{S}_c \triangleq \{(A_c, B_c, C_c) : \tilde{A} \text{ is asymptotically stable}\}$. Assume $(A_c, B_c, C_c) \in \mathcal{S}_c$ and define $\tilde{Q} \in \mathcal{R}^{(n_x+n_c)\times(n_x+n_c)}$ to be the closed-loop steady-state covariance, i.e.,

$$0 = \tilde{A}\tilde{Q} + \tilde{Q}\tilde{A}^T + \tilde{V} \tag{2.12}$$

where

$$\tilde{V} \triangleq \begin{bmatrix} V_1 & V_{12}B_c^T \\ B_cV_{12}^T & B_cV_2B_c^T \end{bmatrix}, \quad V_1 \triangleq {D_1}^T D_1, \quad V_{12} \triangleq 2{D_1}^T D_2, \quad V_2 \triangleq {D_2}^T D_2. \tag{2.13}$$

(Note that since D_2 has full row rank, $V_2 > 0$.) The cost function J can now be expressed as

$$J(A_c, B_c, C_c, \tilde{Q}) = \mathrm{tr}\tilde{Q}\tilde{R}. \tag{2.14}$$

The objective is to minimize the cost function J subject to the constraint (2.12).

The Lagrangian $\mathcal{L}$ is defined by

$$\mathcal{L}(A_c, B_c, C_c, \tilde{Q}, \tilde{P}) \triangleq \mathrm{tr}\tilde{Q}\tilde{R} + \mathrm{tr}[\tilde{P}(\tilde{A}\tilde{Q} + \tilde{Q}\tilde{A}^T + \tilde{V})] \tag{2.15}$$

where $\tilde{P}$ is the Lagrange multiplier matrix. The compensator (A_c, B_c, C_c) is optimal if it satisfies the stationary conditions

$$\frac{\partial\mathcal{L}}{\partial A_c} = 0, \quad \frac{\partial\mathcal{L}}{\partial B_c} = 0, \quad \frac{\partial\mathcal{L}}{\partial C_c} = 0, \tag{2.16}$$

$$\frac{\partial\mathcal{L}}{\partial\tilde{Q}} = \tilde{A}\tilde{P} + \tilde{P}\tilde{A}^T + \tilde{R} = 0, \tag{2.17}$$

and

$$\frac{\partial \mathcal{L}}{\partial \tilde{P}} = \tilde{A}\tilde{Q} + \tilde{Q}\tilde{A}^T + \tilde{V} = 0. \tag{2.18}$$

Both the descent and continuation algorithms aim at finding $(A_c, B_c, C_c) \in \mathcal{S}_c$ that satisfy the above conditions.

Subsequently, we will represent the controller by a parameter vector θ, for example,

$$\theta = \begin{bmatrix} \text{vec}(A_c) \\ \text{vec}(B_c) \\ \text{vec}(C_c) \end{bmatrix}. \tag{2.19}$$

Let the mapping from a state space representation of a controller (A_c, B_c, C_c) to the parameter vector θ be given by $g(\cdot)$, such that

$$\theta = g(A_c, B_c, C_c) \tag{2.20}$$

and define

$$\Theta = \{\theta = g(A_c, B_c, C_c) : (A_c, B_c, C_c) \in S_c \cup \text{dom}(g)\} \tag{2.21}$$

Now, assuming $\theta \in \Theta$, the H_2 cost functional and the corresponding Lagrangian can be expressed respectively as $J(\theta, \tilde{Q})$ and $\mathcal{L}(\theta, \tilde{Q}, \tilde{P})$. The problem is therefore to find $\theta \in \Theta$ such that

$$0 = \frac{\partial \mathcal{L}}{\partial \theta}(\theta, \tilde{Q}, \tilde{P}). \tag{2.22}$$

subject to (2.18) and (2.17).

2.2. Reduction of the Dimension of the Controller Parameter Vector (θ)

It is desired that the parameter vector θ be as small as possible. Hence, we desire to represent the controller matrix with the fewest parameters possible. The minimal number of parameters $p_{\min}$ with which a compensator can be represented is given by (Denery *et al.* 1971, Martin and Bryson 1980)

$$p_{\min} = n_c(m + l). \tag{2.23}$$

One canonical form which allows representation of a controller with a minimal number of parameters is the modal form described in (Martin and Bryson 1980). This form will be called here the Second-Order Polynomial (SP) form. For this parameterization a triple (A_c, B_c, C_c) of order n_c has the following structure.

$$A_c = \text{block-diag}\{A_{c,1}, A_{c,2}, ..., A_{c,n_c}\} \tag{2.24}$$

where $A_{c,i}$ is 2×2, $i = \{1, 2, ..., n_c\}$ and each $A_{c,i}$ (with the exception of A_{c,n_c} if the row dimension of A_c is odd) has the form

$$A_{c,i} = \begin{bmatrix} 0 & 1 \\ a_{c,i}^{(1)} & a_{c,i}^{(2)} \end{bmatrix}, \tag{2.25}$$

to allow for either a complex conjugate set of poles or two real poles. B_c is completely-full and

$$C_c = [C_{c,1}, C_{c,2}, , ..., C_{c,r}], \tag{2.26}$$

where $C_{c,i}$ has the form

$$C_{c,i} = \begin{bmatrix} 1 & 0 \\ * & * \\ \vdots & \vdots \\ * & * \end{bmatrix}. \tag{2.27}$$

The controller canonical form described in (Kailath 1980) also allows representation of a controller with a minimal number of parameters. For SISO systems in controller canonical form the A_c matrix is a companion matrix. In particular, A_c has the form

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & 0 \\ * & * & * & * & \cdots & * \end{bmatrix}. \tag{2.28}$$

In addition,

$$B_c = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \tag{2.29}$$

and C_c is completely full. A dual form of the controller canonical form is the observable canonical form (Kailath 1980).

Another minimal parameter basis is the "input normal Riccati form" of (Davis *et al.* 1994). In this basis A_c is completely determined by B_c and C_c.

It is also possible to represent the controller in a basis where the number of free parameters p satisfies

$$p_{\min} < p < p_{\max} \triangleq n_c (n_c + m + l). \tag{2.30}$$

One such basis is the tridiagonal basis (Geist 1991, Parlett 1992) in which the controller state matrix is constrained to have nonzero elements only on the diagonal, the super-diagonal, and the

sub-diagonal. That is,

$$A_c = \begin{bmatrix} * & * & & & & \\ * & * & * & & 0 & \\ & * & * & & \ddots & \\ & 0 & \ddots & \ddots & & \\ & & & & * & * \end{bmatrix}, \tag{2.31}$$

and B_c and C_c are completely full. For this form the number of free parameters is given by

$$p = p_{\min} + (3n_c - 2)$$

It is important to recognize that, given a particular basis, there are stabilizing controllers that have no state space realization in that basis, such that if $g(\cdot)$ in (2.20) requires the representation of a controller in a given basis $S_c \cup \mathrm{dom}(g) \subset S_c$. Hence, the set Θ, defined by (2.21) corresponds to a smaller set of controllers than the set S_c. When the controller is restricted to a particular basis in the algorithms to follow, this reduction in the size of the feasible set sometimes leads to numerical ill-conditioning or even algorithm failure.

3. Parameter Optimization Algorithms

This section first gives a general description of the algorithms corresponding to the descent methods. It then briefly describes a continuation algorithm. Particular attention is given to the modification of these algorithms to take into account the constraint $\theta \in \Theta$.

3.1. Descent Methods

Descent methods are designed to search for solutions to the unconstrained optimization problem

$$\min_{\theta} J(\theta). \tag{3.1}$$

The user is required to supply an initial parameter vector θ_0. A descent algorithm then has the following structure.

A Descent Algorithm

1. Let $k = 0$.
2. Determine a search direction d_k.
3. Use a one dimensional line search to find α_k that minimizes $J(\theta_k + \alpha d_k)$ with respect to α.

4. Set $\theta_{k+1} = \theta_k + \alpha_k d_k$

5. If the gradient $\frac{\partial J}{\partial \theta}(\theta_{k+1})$ is sufficiently small, then let the optimal solution $\theta^* = \theta_{k+1}$ and stop, else let $k = k + 1$ and go to Step 2.

Alternative descent methods differ primarily in the way they compute the descent direction d_k. For example, in the steepest descent method d_k corresponds to the negative of the gradient. Conjugate gradient and Quasi-Newton methods compute d_k using only cost and gradient information while Newton's method requires computation of the Hessian matrix. Note that for the H_2 optimal, reduced-order control problem it is not difficult to show that if (2.18) and (2.17) are satisfied, then the gradient satisfies

$$\frac{\partial J}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \theta}. \tag{3.2}$$

Hence, the gradient may be computed by constructing and differentiating the Lagrangian.

Recognize that the H_2 optimal, reduced-order control problem is not the unconstrained optimization problem (3.1) but is actually the constrained optimization problem

$$\min_{\theta \in \Theta} J(\theta) \tag{3.3}$$

where Θ is defined by (2.21). One way to take into account the constraint $\theta \in \Theta$ is to modify the line search subalgorithm of Step 3 to ensure that if $\theta_k \in \Theta$, θ_{k+1} is also in Θ. (It is assumed that $\theta_0 \in \Theta$.) An example of such a modification is discussed in (Kuhn and Shmidt 1987). The descent algorithms compared in this paper use the following modification to the line search algorithm of (Fletcher 1987). During the bracketing phase of the line search algorithm a stability check is carried out to make sure that the right end of the bracket results in a stable closed-loop system. If not, then the bracket is halved. This simple modification ensures that all points within the bracket result in a stable closed-loop system.

3.2. Continuation Methods

Continuation techniques can be used to solve the zero finding problem

$$0 = f(\theta), \tag{3.4}$$

where $f : \mathcal{R}^p \to \mathcal{R}^p$. In the context of H_2 optimal, reduced-order control, (3.4) corresponds to (2.22). Continuation techniques work by finding a C^2 function $H : \mathcal{R}^p \times [0, 1) \to \mathcal{R}^p$ that satisfies certain properties, including the following:

1. $H(\theta, 1) = f(\theta)$;

2. $0 = H(\theta, 0)$ has an easily found or known solution θ_0.

They then trace the zero curve described by

$$0 = H(\theta, \lambda), \quad \lambda \in [0, 1). \tag{3.5}$$

This is accomplished by differentiating (3.5) with respect to λ to obtain Davidenko's differential equation

$$0 = H_\lambda(\theta, \lambda) + H_\theta(\theta, \lambda)\theta_\lambda(\lambda) \tag{3.6}$$

where $H_\lambda \triangleq \frac{\partial H}{\partial \lambda}$, $H_\theta \triangleq \frac{\partial H}{\partial \theta}$, and $\theta_\lambda \triangleq \frac{d\theta}{d\lambda}$, which together with $\theta(0) = \theta_0$ defines an initial value problem. Predictor-corrector, numerical integration schemes are then used to solve this initial value problem, that is to follow the curve (3.5) from the solution θ_0 of $0 = H(\theta, 0)$ to a solution θ^* of $0 = H(\theta, 1)$. In particular a continuation algorithm has the following structure.

A Continuation Algorithm

1. Let $\lambda = 0$ and $\theta(\lambda) = \theta_0$.

2. Use (3.6) to compute the tangent vector θ_λ, such that $\theta_\lambda(\lambda) = -H_\theta(\theta, \lambda)^{-1} H_\lambda(\theta, \lambda)$.

3. For some $\Delta\lambda$ such that $\lambda = \lambda + \Delta\lambda \leq 1$, use current and past values of H and H_λ to predict $\theta(\lambda + \Delta\lambda)$ by using polynomial curve fitting.

4. Let $\lambda \leftarrow \lambda + \Delta\lambda$ and θ_0 be the prediction of $\theta(\lambda)$.

5. For $k = 0, 1, 2, \ldots$ until convergence, do

$$\theta_{k+1} = \theta_k - H_\theta(\theta_k, \lambda)^{-1}\theta_k.$$

   Then, let $\theta(\lambda) = \theta_{k+1}$.

6. If $\lambda < 1$, go to Step 2, else if $\lambda = 1$, then let the solution $\theta^* = \theta(\lambda)$ and stop.

The initializing controller θ_0 in the algorithm for H_2 optimal, reduced-order control (Collins *et al.* 1994, Collins *et al.* 1996a) is usually found by applying a controller reduction method such as balanced controller reduction (Yousuff and Skelton 1984a) to a low authority LQG controller since this usually yields a nearly optimal, reduced-order controller. The initial weights $(R_1)_0$, $(R_{12})_0$, $(R_2)_0$, $(V_1)_0$, $(V_{12})_0$, $(V_2)_0$ corresponding to the low authority LQG controller are

then deformed into the desired weights along the homotopy path. The reader is referred to (Collins *et al.* 1994) for further details.

The algorithm of (Collins *et al.* 1994) also assumes that the prediction $\theta(\lambda + \Delta\lambda) \in \Theta$ such that it corresponds to a controller that stabilizes the closed-loop system. If $\theta(\lambda + \Delta\lambda) \notin \Theta$, then the algorithm reduces the size of $\Delta\lambda$. In particular, $\Delta\lambda \leftarrow \frac{1}{2}\Delta\lambda$. This is a small but necessary modification. As subsequently discussed, it is at this point that the algorithm sometimes fails.

4. Numerical Examples

4.1. Description of Problems

The first problem is a noncollocated axial vibration control problem involving an axial beam with four circular disks attached. This problem was introduced in (Cannon and Rosenthal 1984) and also studied in (Collins *et al* 1994). The plant is 8th order while we consider the design of a 4th order controller.

The second problem was introduced in (Ly *et al.* 1985) and involves flight control for a NAVION aircraft. The model is 7th order and we consider the design of a 4th order controller.

The third problem was introduced in (Martin and Bryson 1980) and involves vibration control of a flexible spacecraft. The model is 6th order while we again consider the design of a 4th order controller.

The system matrixes (A, B, C, D) and the weighting matrices $(R_1, R_2, R_{12}, V_1, V_2, V_{12})$ are given in the Appendix. Note that the matrices R_2 and V_2 are multiplied by ρ which is allowed to change from 10 to 1 in order to deform the low authority controller to a higher authority controller using the homotopy algorithm. In the case of the descent algorithms ρ is fixed at 1.

For each example, a low authority optimal LQG controller (corresponding to $\rho = 10$) was first designed. The order of this controller was then reduced using the modified balanced controller reduction technique of (Yousuff and Skelton 1984a). This reduced order sub-optimal controller was then converted into an optimal low authority controller using a few Newton iterations. This controller was used as the starting point for both the continuation and descent optimization methods. These numerical examples are included in Tables 1 through 3 and were run on a 90 MHz, Pentium PC. Table 4, which only presents data for the continuation algorithm, was produced using a 120 MHz Pentium PC.

We also design higher order controllers for all three examples using continuation algorithms with the controller unconstrained and with the controller constrained to the tridiagonal basis, in

order to compare the two methods. These numerical examples are included in Table 4 and were run on a 120 MHz Pentium PC.

Method/Basis	Unconstrained		Tridiagonal		SPF		CCF	
	M Flops	Time (sec)	M Flops	Time (sec)	M Flops	Time (sec)	M Flops	Time (sec)
Continuation	19.8	55	14.8	50.8	374	1422	349.8	1987
Newton	13.85	41.1	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls
BFGS	13.45	41.3	3.4	15.3	f-ls	f-ls	f-ls	f-ls
Conjugate Grad.	56.5	167.5	13.7	46.8	f-osc	f-osc	f-ls	f-ls
Steepest Des.	227	727	68.3	207.3	f-osc	f-osc	f-osc	f-osc

Table 1: FOUR DISKS

4.2. Observations

Comparison of the various algorithms are given in Tables 1, 2, and 3. The letter "f" denotes failure of an algorithm. In particular "f-ls" denotes failure due to the step length parameter computed by the line search, becoming extremely small. This occurs because of the modification to the line search algorithm, to make sure that the controller results in a stable closed-loop. During the line search if the resulting controller violates this constraint, as previously described, the step size is reduced by half. This sometimes results in extremely small step sizes for which there is no appreciable decrease in cost or change in the search direction. Hence the algorithm effectively comes to a stop.

Similarly, "f-λ" denotes failure due to the need for extremely small increments in the homotopy parameter λ. This occurs because of the modification to the prediction step in the continuation algorithms to avoid unstable closed-loops. If during any of these steps, an unstable closed loop is obtained, then, as previously described, the increment to the continuation parameter λ, is decreased by half. This sometimes results in the need for extremely small increments in the homotopy parameter λ for which there is no appreciable change in the controller parameters θ.

The indicator "f-osc" denotes failure in a descent algorithm, due to oscillatory behavior of the gradient of the cost function as the algorithm progresses. Several hundreds of iterations are performed without any perceptible change in the cost function. This phenomena was observed primarily in the steepest descent algorithm and is a well known deficiency of this method (Fletcher 1987).

Tables 1 through 3 reveal that constraining the basis of the controller led to increased failure in

Method/Basis	Unconstrained		Tridiagonal		SPF		CCF	
	M Flops	Time (sec)	M Flops	Time (sec)	M Flops	Time (sec)	M Flops	Time (sec)
Continuation	537	210	470	210.2	f-λ	f-λ	f-λ	f-λ
Newton	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls
BFGS	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls
Conjugate Grad.	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls
Steepest Des.	f-osc	f-osc	f-osc	f-osc	f-ls	f-ls	f-ls	f-ls

Table 2: NAVION

Method/Basis	Unconstrained		Tridiagonal		SPF		CCF	
	M Flops	Time (sec)	M Flops	Time (sec)	M Flops	Time (sec)	M Flops	Time (sec)
Continuation	12.6	10.38	21.7	21.1	25.1	23.4	121.8	94.85
Newton	11.7	9.94	5.9	9.88	5.22	7.25	7.57	8.9
BFGS	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls	f-ls
Conjugate Grad.	35.88	25.3	253.4	159.8	74.8	51.19	517.7	302.6
Steepest Des.	200.8	147.2	f-osc	f-osc	5.66	6.92	285	193

Table 3: SPACECRAFT

Example(Controller Order)/Basis	Unconstrained		Tridiagonal	
	M Flops	Time (sec)	M Flops	Time (sec)
Fourdisks (6^{th} order)	1028.5	294.5	412.8	164
Fourdisks (8^{th} order)	3192.7	596	809.9	239.4
Navion (6^{th} order)	2815	600.8	1566.2	457.7
Spacecraft (6^{th} order)	347.9	130.1	196.5	98.4

Table 4: Continuation: Unconstrained vs Tridiagonal

both the continuation and descent algorithms (i.e. by both the "f-λ" and the "f-ls" failure modes). One apparent reason for this is that constraining the controller to a particular basis reduces the feasible subspace, as discussed at the end of Section 2. Algorithm failure due to basis constraints was also observed in (Kuhn and Schmidt 1987).

The numerical conditioning of the algorithms when using the tridiagonal basis was better than when using the second order polynomial form (SPF) and the controller canonical form (CCF) and is apparently due to the fact that the tridiagonal form is a more general representation than SPF and CCF. In fact, SPF is a special case of a tridiagonal form. This phenomena can be seen by a comparison of execution times for the respective bases in Tables 1 through 3. For most cases, the execution times for the SPF and the CCF bases, are several times larger than the those for the tridiagonal basis, even though the SPF and CCF are minimal parameter bases, while the tridiagonal is not, and hence have smaller parameter vectors θ. This is due to the decreased numerical conditioning associated with the use of the SPF and CCF basis.

The conjugate gradient method, as might be expected, usually converged faster than the steepest descent method. The convergence times for the Newton, BFGS and continuation methods were equivalent and usually much less than the convergence times for the conjugate gradient and the steepest descent methods. However all the descent methods including Newton and BFGS methods fail much more often than did the continuation methods. This is especially true when the controller basis is unconstrained or tridiagonal, in which case the continuation method never failed.

In Tables 1 through 3, it is seen that the run times of the continuation algorithm for the unconstrained and tridiagonal cases were very similar. However, as the controller dimension increases, the size of the parameter vector associated with the unconstrained "basis" increases much more rapidly than the parameter vector associated with the tridiagonal basis. Hence, it is expected that the convergence times for the tridiagonal case will increase much less rapidly than the unconstrained case as the controller dimension increases. This is confirmed in Table 4 which was generated using a 120 MHz Pentium PC.

5. Conclusions

This paper has used three examples to compare the behavior of four standard descent algorithms with a recently developed continuation algorithm for H_2 optimal, reduced-order design. The results clearly indicate that the continuation algorithm is much more numerically robust than any of the descent algorithms, especially when the controller basis is unconstrained or tridiagonal. The numerical conditioning of the algorithms always decreased when the controller was constrained

to either the second order polynomial form or the controller canonical form which are minimal parameter bases. However numerical conditioning of the algorithm when using the tridiagonal form, a nonminimal parameter basis, was better than when using a minimal parameter basis. The numerical conditioning of the continuation algorithms, when using the tridiagonal basis was unchanged or at worst marginally decreased, as compared to the unconstrained case. Hence, when using the tridiagonal basis, the advantage of a smaller parameter vector θ, usually outweighed the disadvantage of reduced numerical conditioning due to a basis constraint. This advantage, as expected, became more apparent as the order of the controller was increased.

References

[1] B. D. O. Anderson and Y. Liu, 1989, Controller reduction: concepts and approaches. *IEEE Transactions on Automatic Control*, **34**, 802-812.

[2] R. H. Cannon and D. E. Rosenthal, 1984, Experiments in control of flexible structures with noncolocated sensors and actuators. *Journal of Guidance, Control and Dynamics*, **7**, 546-553.

[3] E. G. Collins, Jr., L. D. Davis and S. Richter, 1994, Design of reduced-order, H_2 optimal control using a homotopy algorithm. *International Journal of Control*, **61**, 849-852.

[4] E. G. Collins, Jr., W. M. Haddad and S. S. Ying, 1996a, Nearly nonminimal Linear-Quadratic-Gaussian compensators for reduced-order control design initialization. *Journal of Guidance, Control and Dynamics*, **19**, 259-261.

[5] E. G. Collins, Jr., S. S. Ying, and W. M. Haddad, 1996b, Reduced-order dynamic compensation using the Hyland and Bernstein optimal projection equations. *Journal of Guidance, Control, and Dynamics*, to appear.

[6] L. D. Davis, E. G. Collins, Jr., and A. S. Hodel, 1994, A parameterization of minimal plants. *IEEE Transactions on Automatic Control*, **39**, 849-852.

[7] D. G. Denery, 1971, An identification algorithm that is insensitive to initial parameter estimates. *AIAA Journal*, **9**, 371-377.

[8] C. De Villemagne and R. E. Skelton, 1988, Controller reduction using canonical interactions. *IEEE Transactions on Automatic Control*, **33**, 740-750.

[9] R. Fletcher, 1987, *Practical methods of optimization*: Second Edition, John Wiley and Sons, New York.

[10] G. A. Geist, 1991, Reduction of a general matrix to tridiagonal Form. *SIAM Journal of Matrix Analysis and Applications*, **12**, 362-373.

[11] T. Kailath, 1980, *Linear systems*, Prentice-Hall, New Jersey.

[12] F. S. Kramer and A. J. Calise, 1987, Fixed-order dynamic compensation for multivariable linear systems. *Journal of Guidance and Control*, **11**, 80-85.

[13] U. Kuhn and G. Schmidt, 1987, Fresh look into the design and computation of optimal output feedback controls for linear multivariable systems. *International Journal of Control*, **46**, 75-95, 1987.

[14] H. Kwakernaak and R. Sivan, 1972, *Linear optimal control Systems*, Wiley-Interscience.

[15] Y. Liu, B. D. O. Anderson and U. L. Ly, 1990, Coprime factorization controller reduction with Bezout identity induced frequency weighting. *Automatica*, **26**, 233-249.

[16] U. L. Ly, A. E. Bryson and R. H. Cannon, 1985, Design of low-order compensators using parameter optimization. *Automatica*, **21**, 315-318.

[17] G. D. Martin and A. E. Bryson, Jr., 1980, Attitude control of a flexible spacecraft. *Journal of Guidance, Control and Dynamics*, **3**, 37-41.

[18] M. Mercadal, 1991, Homotopy approach to optimal, linear quadratic, fixed architecture compensation. *Journal of Guidance, Control and Dynamics*, **14**, 1224-1233.

[19] V. Mukhopadhyay, J. R. Newsom, and I. Abel, 1982, Reduced-order optimal feedback control law for flutter suppression. *Journal of Guidance and Control*, **5**, 389-395.

[20] V. Mukhopadhyay, 1987, Stability robustness improvement using constrained optimization techniques. *Journal of Guidance and Control*, **10**, 172-177.

[21] V. Mukhopadhyay, 1989, Digital robust control law synthesis using constrained optimization. *Journal of Guidance and Control*, **12**, 175-181, 1989.

[22] B. N. Parlett, 1992, Reduction to tridiagonal form and minimal realization. *SIAM Journal of Matrix Analysis and Applications*, **12**, 567-593.

[23] C. Voth and U. L. Ly, 1991, Design of a total energy control autopilot using constrained parameter optimization. *Journal of Guidance, Control and Dynamics*, **14**, 927-935.

[24] A. Yousuff and R. E. Skelton, 1984a, A note on balanced controller reduction. *IEEE Transactions on Automatic Control*, **29**, 254-257.

[25] A. Yousuff and R. E. Skelton, 1984b, Controller reduction by component cost analysis. *IEEE Transactions on Automatic Control*, **29**, 520-530.

APPENDIX

The state space descriptions of the 3 examples are as follows.

AXIAL VIBRATION PROBLEM

$$A = \begin{bmatrix} -0.0370 & 1.8496 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1.8496 & -0.0370 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.0282 & 1.4097 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1.4097 & -0.0282 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.0153 & 0.7648 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.7648 & -0.0153 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B^T = \begin{bmatrix} -0.1497 & 0.5691 & -0.4961 & -1.2918 & -2.1635 & -1.3608 & 0.1045 & 0.9955 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.1526 & 0.0426 & 0.1181 & -0.0434 & 0.0481 & -0.0768 & 0.2511 & -0.0257 \end{bmatrix}, \quad D = 0$$

$$R_1 = 1.0e{-}003 \begin{bmatrix} 0.0007 & -0.0031 & -0.0013 & -0.0030 & -0.0039 & -0.0020 & -0.0011 & -0.0152 \\ -0.0031 & 0.0135 & 0.0057 & 0.0132 & 0.0171 & 0.0088 & 0.0048 & 0.0661 \\ -0.0013 & 0.0057 & 0.0024 & 0.0056 & 0.0072 & 0.0037 & 0.0021 & 0.0280 \\ -0.0030 & 0.0132 & 0.0056 & 0.0130 & 0.0167 & 0.0086 & 0.0048 & 0.0649 \\ -0.0039 & 0.0171 & 0.0072 & 0.0167 & 0.0215 & 0.0111 & 0.0061 & 0.0835 \\ -0.0020 & 0.0088 & 0.0037 & 0.0086 & 0.0111 & 0.0057 & 0.0031 & 0.0429 \\ -0.0011 & 0.0048 & 0.0021 & 0.0048 & 0.0061 & 0.0031 & 0.0017 & 0.0238 \\ -0.0152 & 0.0661 & 0.0280 & 0.0649 & 0.0835 & 0.0429 & 0.0238 & 0.3240 \end{bmatrix}$$

$$R_2 = \rho, \quad {R_{12}}^T = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0], \quad V_2 = \rho, \quad {V_{12}}^T = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0].$$

$$V_1 = \begin{bmatrix} 0.0368 & -0.1560 & 0.2049 & 0.4960 & -0.5984 & -0.3537 & -0.0200 & -0.1909 \\ -0.1560 & 0.6615 & -0.8690 & -2.1031 & 2.5374 & 1.4997 & 0.0850 & 0.8097 \\ 0.2049 & -0.8690 & 1.1417 & 2.7630 & -3.3335 & -1.9703 & -0.1117 & -1.0637 \\ 0.4960 & -2.1031 & 2.7630 & 6.6866 & -8.0673 & -4.7682 & -0.2702 & -2.5742 \\ -0.5984 & 2.5374 & -3.3335 & -8.0673 & 9.7332 & 5.7529 & 0.3260 & 3.1058 \\ -0.3537 & 1.4997 & -1.9703 & -4.7682 & 5.7529 & 3.4002 & 0.1927 & 1.8357 \\ -0.0200 & 0.0850 & -0.1117 & -0.2702 & 0.3260 & 0.1927 & 0.0109 & 0.1040 \\ -0.1909 & 0.8097 & -1.0637 & -2.5742 & 3.1058 & 1.8357 & 0.1040 & 0.9910 \end{bmatrix}$$

$$A = \begin{bmatrix} -0.0450 & 0.0360 & 0 & -0.3220 & 0 & 0.0450 & -0.0360 \\ -0.3700 & -2.0200 & 1.7600 & 0 & 0 & 0.3700 & 2.0200 \\ 0.1910 & -3.9600 & -2.9800 & 0 & 0 & -0.1910 & 3.9600 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & -1.0000 & 0 & 1.7600 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.4820 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.0570 \end{bmatrix}$$

$$B^T = \begin{bmatrix} 0 & -0.2820 & -11.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad D^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$R_1 = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 & -1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0625 & 0 & 0 \\ -1.0000 & 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R_2 = \rho \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix}, \quad {R_{12}}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

$$V_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 111.9343 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 111.9261 \end{bmatrix}$$

$$V_2 = \rho \begin{bmatrix} 0.1600 & 0 & 0 \\ 0 & 0.1600 & 0 \\ 0 & 0 & 100.0000 \end{bmatrix}, \quad {V_{12}}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

SPACECRAFT

$$A = \begin{bmatrix} -0.0836 & -2.1904 & 0 & 0 & 0 & 0 \\ 2.1904 & 0 & 0 & 0 & 0 & 0 \\ -0.0738 & -1.7519 & -0.0107 & -0.8710 & 0 & 0 \\ 0 & 0 & 0.8710 & 0 & 0 & 0 \\ -0.0738 & -1.7519 & -0.0063 & -0.6487 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \end{bmatrix}$$

$$B^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -0.0066 \end{bmatrix}, \quad D = 0$$

$$R_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.4290 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.4290 \end{bmatrix}, \quad R_2 = \rho, \quad R_{12} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

$$V_1 = 1.0e-006 \begin{bmatrix} 1.0000 & 0 & 1.0000 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1.0000 & 0 & 1.0000 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1.0000 & 0 & 1.0000 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad V_2 = 2.1000e-008\,\rho, \quad V_{12} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Appendix C

"Cost-Effective Parallel Processing for H_2/H_∞ Controller Synthesis"

Cost-Effective Parallel Processing for H^2/ H^∞ Controller Synthesis

Yuzhen Ge† Layne T. Watson‡ Emmanuel G. Collins, Jr.*

Abstract

A distributed version of a homotopy algorithm for solving the H^2/H^∞ mixed-norm controller synthesis problem is presented. The main purpose of the study is to explore the possibility of achieving high performance with low cost. Existing UNIX workstations running PVM (Parallel Virtual Machine) are utilized. Only the Jacobian matrix computation is distributed and therefore the modification to the original sequential code is minimal. The same algorithm has also been implemented on an Intel Paragon parallel machine. Our implementation shows that acceptable speedup is achieved and the larger the problem sizes, the higher the speedup. Comparing with the results from the Intel Paragon, the study concludes that utilizing the existing UNIX workstations can be a very cost-effective approach to shorten computation time. Furthermore, this economical way to achieve high performance computation can easily be realized and incorporated in a practical industrial design environment.

1 Introduction

H^2/H^∞ mixed-norm controller synthesis is an important and interesting technique in modern control design which provides the means for simultaneously addressing H^2 and H^∞ performance objectives. In practice such controllers provide both nominal performance (via suboptimal H^2) and robust stability (via H^∞). Hence mixed-norm synthesis provides a technique for trading off performance and robustness, a fundamental objective in control design.

The H^2/H^∞ mixed-norm problem has been addressed in a variety of settings. One treatment utilized an H^2 cost bound as the basis for an auxiliary nonconvex constrained minimization problem, which is very difficult to solve without the global convergence of homotopy methods. A successful homotopy algorithm based on the Ly form parametrization has been developed (Ge *et al.* 1994).

The H^2/H^∞ control design algorithms will be used for controller design of systems such as the four disk system of (Cannon and Rosenthal 1984). This system is especially representative of the type of vibration control problems that arise in industrial problems involving rotating turbomachinery. H^2/H^∞ design will be used to develop controllers that

The work of Yuzhen Ge and Emmanuel G. Collins, Jr. was supported in part by Air Force Office of Scientific Research grant F49620-95-1-0244 and the work of Layne T. Watson was supported in part by Air Force Office of Scientific Research grant F49620-92-J-0236 and Department of Energy grant DE-FG05-88ER25068/A004.

† Department of Mathematics and Computer Science, Butler University, Indianapolis, IN 46208. ge@butler.edu.

‡ Departments of Computer Science and Mathematics, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061-0106. ltw@cs.vt.edu.

* Department of Mechanical Engineering, Florida A&M/Florida State University, Tallahassee, FL 32310. ecollins@eng.fsu.edu.

are robust with respect to unmodeled dynamics and also guarantee a certain measure of nominal performance.

It should be mentioned that H^2/H^∞ theory has been used in (Haddad *et al.* 1993)–(Haddad *et al.* 1994b) to develop complex structured singular value synthesis (CSSV) formalisms that *a priori* fix the structure of both the D-scales and the controller. Hence, an extension of the algorithms here will enable fixed-structure CSSV controller synthesis that blends H^2 and H^∞ performance objectives.

Practical applications often lead to large dense systems of nonlinear equations which are time-consuming to solve on a serial computer. For these systems, parallel processing may be the only feasible means to achieving solution algorithms with acceptable speed. One economical way of achieving parallelism is to utilize the aggregate power of a network of heterogeneous serial computers. In industrial environments where interactive design is often the practice, the parallel code can be easily incorporated into interactive software such as MATLAB or Mathematica with proper setup of the network computers. To the engineering users the design environment is identical. However the computations are faster.

The most expensive part of the H^2/H^∞ homotopy algorithm of (Ge *et al.* 1994) is the computation of the Jacobian matrix, which can be parallelized easily to run across an Ethernet network with little modification of the original sequential code, and which has relatively large task granularity. There is a trade-off between the programming effort and the speedup of the parallel program. To obtain a better speedup, other parts of the homotopy algorithm, such as finding the solution to the Riccati equations and the QR factorization to compute the kernel of the Jacobian matrix, need to be parallelized as well.

In this study the homotopy algorithm for H^2/H^∞ controller synthesis is parallelized to run on a network of workstations using PVM (Parallel Virtual Machine) and on an Intel Paragon parallel computer, under the philosophy that as few changes as possible are to be made to the sequential code while achieving an acceptable speedup. The parallelized computation is that of the Jacobian matrix, which is carried out in the master-slave paradigm by functional parallelism, that is, each machine computes a different column of the Jacobian matrix with its own data. Unless the Riccati equation solver is parallelized, there is a large amount of data needed for each slave process at each step of the homotopy algorithm. To avoid sending too many large messages across the network or among different nodes on the Intel Paragon, all slave processes repeat part of the computation done by the master process, which therefore decreases the speedup of the parallel computation.

The speedups of the parallel code are compared as the number of workstations increases or as the number of nodes increases on an Intel Paragon and as the size of the problem varies. A reasonable speedup can be achieved using an existing network of workstations compared to that of using an expensive parallel machine, the Intel Paragon. It is demonstrated that for a large problem, the approach of using a network of workstations to parallelism is feasible and practical, and provides an efficient and economical computational method to parallelize a homotopy based algorithm for H^2/H^∞ controller synthesis in a workstation-based interactive design environment.

2 The H^2/H^∞ controller synthesis problem

The LQG controller synthesis problem with an H^∞ performance bound can be stated as follows: given the n-th order stabilizable and detectable plant

$$\begin{aligned} \dot{x}(t) &= A\,x(t) + B\,u(t) + D_1 w(t), \\ y(t) &= C\,x(t) + D_2 w(t), \end{aligned} \tag{1}$$

where $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{n\times m}$, $C \in \mathbf{R}^{l\times n}$, $D_1 \in \mathbf{R}^{n\times p}$, $D_2 \in \mathbf{R}^{l\times p}$, $D_1 D_2^T = 0$, and $w(t)$ is p-dimensional white noise, find a n_c-th order dynamic compensator

$$\begin{aligned} \dot{x}_c(t) &= A_c\,x_c(t) + B_c\,y(t), \\ u(t) &= C_c\,x_c(t), \end{aligned} \tag{2}$$

where $A_c \in \mathbf{R}^{n_c\times n_c}$, $B_c \in \mathbf{R}^{n_c\times l}$, $C_c \in \mathbf{R}^{m\times n_c}$, and $n_c \le n$, which satisfies the following criteria:

(i) the closed-loop system (1)–(2) is asymptotically stable, i.e., $\tilde{A} = \begin{pmatrix} A & B\,C_c \\ B_c\,C & A_c \end{pmatrix}$ is asymptotically stable;

(ii) the $q_\infty \times p$ closed-loop transfer function

$$H(s) \equiv \tilde{E}_\infty (sI_{\tilde{n}} - \tilde{A})^{-1} \tilde{D},$$

from $w(t)$ to

$$z(t) = E_{1\infty} x(t) + E_{2\infty} u(t), \tag{3}$$

where $\tilde{E}_\infty = (\,E_{1\infty} \quad E_{2\infty} C_c\,)$ ($E_{1\infty} \in \mathbf{R}^{q_\infty\times n}$, $E_{2\infty} \in \mathbf{R}^{q_\infty\times m}$, $E_{1\infty}^T E_{2\infty} = 0$), $\tilde{n} = n + n_c$, and $\tilde{D} = \begin{pmatrix} D_1 \\ B_c D_2 \end{pmatrix}$, satisfies the constraint

$$\|H(s)\|_\infty \le \gamma \tag{4}$$

where $\gamma > 0$ is a given constant; and

(iii) the performance functional

$$J(A_c, B_c, C_c) \equiv \lim_{t\to\infty} \mathcal{E}\left[x^T(t) R_1 x(t) + u^T(t) R_2 u(t)\right] \tag{5}$$

is minimized, where $\mathcal{E}$ is the expected value, $R_1 = E_1^T E_1 \in \mathbf{R}^{n\times n}$ and $R_2 = E_2^T E_2 \in \mathbf{R}^{m\times m}$ ($E_1 \in \mathbf{R}^{q\times n}$, $E_2 \in \mathbf{R}^{q\times m}$, $E_1^T E_2 = 0$) are, respectively, symmetric positive semidefinite and symmetric positive definite weighting matrices.

The closed-loop system (1)–(3) can be written as the augmented system

$$\begin{aligned} \dot{\tilde{x}}(t) &= \tilde{A}\,\tilde{x}(t) + \tilde{D}\,w(t), \\ z(t) &= \tilde{E}_\infty \tilde{x}(t) \end{aligned} \tag{6}$$

where $\tilde{x} = \begin{pmatrix} x \\ x_c \end{pmatrix}$.

Using this notation and under the condition that $\tilde{A}$ is asymptotically stable, for a given compensator the performance (5) is given by

$$J(A_c, B_c, C_c) = \mathrm{tr}\,(\tilde{Q}\,\tilde{R}), \tag{7}$$

where $\tilde{R} = \begin{pmatrix} R_1 & 0 \\ 0 & C_c^T R_2 C_c \end{pmatrix}$ and $\tilde{Q}$ satisfies the Lyapunov equation

$$\tilde{A}\,\tilde{Q} + \tilde{Q}\,\tilde{A}^T + \tilde{V} = 0, \tag{8}$$

with symmetric positive semidefinite $V_1 = D_1 D_1^T$, symmetric positive definite $V_2 = D_2 D_2^T$, and

$$\tilde{V} = \begin{pmatrix} V_1 & 0 \\ 0 & B_c V_2 B_c^T \end{pmatrix}.$$

LEMMA 1 (BERNSTEIN AND HADDAD 1989), (HADDAD AND BERNSTEIN 1990). *Let* (A_c, B_c, C_c) *be given and assume there exists* $\mathcal{Q} \in \mathbf{R}^{\tilde{n}\times\tilde{n}}$ *satisfying*

$$\mathcal{Q} \quad \text{is symmetric and nonnegative definite} \tag{9}$$

and

$$\tilde{A}\mathcal{Q} + \mathcal{Q}\tilde{A}^T + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty\mathcal{Q} + \tilde{V} = 0, \tag{10}$$

where $\tilde{R}_\infty = \begin{pmatrix} R_{1\infty} & 0 \\ 0 & C_c^T R_{2\infty} C_c \end{pmatrix}$, $R_{1\infty} = E_{1\infty}^T E_{1\infty}$, *and* $R_{2\infty} = E_{2\infty}^T E_{2\infty}$ *are symmetric positive semidefinite matrices. Then*

$$(\tilde{A}, \tilde{D}) \quad \text{is stabilizable} \tag{11}$$

if and only if

$$\tilde{A} \quad \text{is asymptotically stable.}$$

In this case

$$\|H(s)\|_\infty \le \gamma, \tag{12}$$

$$\tilde{Q} \le \mathcal{Q} \qquad (\mathcal{Q} - \tilde{Q} \text{ is nonnegative definite}),$$

and

$$\text{tr } \tilde{Q}\tilde{R} \equiv J(A_c, B_c, C_c) \le \mathcal{J}(A_c, B_c, C_c) \equiv \text{ tr } \mathcal{Q}\tilde{R}.$$

Hence the satisfaction of (9) and (10) along with the generic condition (11) leads to: 1) closed-loop stability; 2) prespecified H^∞ attenuation; and 3) an upper bound for the H^2 performance criterion.

The auxiliary minimization problem is to determine $(A_c,\ B_c,\ C_c)$ that minimizes $\mathcal{J}(A_c, B_c, C_c)$, and thus provides a bound for the actual H^2 criterion $J(A_c, B_c, C_c)$.

For technical reasons $(A_c, B_c, C_c, \mathcal{Q})$ is restricted to the open set

$$\mathcal{S} \equiv \{(A_c, B_c, C_c, \mathcal{Q}) : \tilde{A} \text{ and } \tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R} \text{ are asymptotically stable,}$$
$$\mathcal{Q} \text{ is symmetric positive definite, and } (A_c, B_c, C_c) \text{ is controllable and observable}\}.$$

3 The homotopy algorithm

Treating all the components of A_c, B_c, C_c as unknowns is redundant, since only $n_c(m+l)$ parameters suffice to describe the full or reduced order controller. There are numerous ways to parametrize the model (A_c, B_c, C_c) — the parametrization introduced by (Ly *et al.* 1985) is one way that uses the minimal number of parameters. In this basis, A_c is a 2×2 block-diagonal matrix (2×2 blocks with an additional 1×1 block if n_c is odd) with 2×2 blocks in the form $\begin{pmatrix} 0 & 1 \\ * & * \end{pmatrix}$, B_c is a full matrix, and C_c has its first row fixed as $(1\ \ 0\ \ 1\ \ 0\ \ \cdots)$. Observe that the Ly structure has $n_c(m+l)$ unknowns — n_c from A_c, $n_c l$ from B_c, and $n_c(m-1)$ from C_c. Denote the unknowns by

$$\theta \equiv \begin{pmatrix} (A_c)_{\mathcal{I}} \\ \text{Vec } (B_c) \\ \text{Vec } (C_c)_{\mathcal{I}^\cdot} \end{pmatrix},$$

where $(A_c)_{\mathcal{I}}$ is a vector consisting of those elements in A_c indexed by the set $\mathcal{I} \equiv \{(2,1),(2,2),\ldots,(n_c,n_c)\}$, Vec of a matrix is the concatenation of its columns, and $(C_c)_{\mathcal{T}}$. is the matrix obtained from rows $\mathcal{T} \equiv \{2,\ldots,m\}$ of C_c.

Choose a problem for which a solution θ_0 is known, defined by the matrices A_0, B_0, C_0, $R_{1,0}$, $R_{2,0}$, $R_{1\infty,0}$, $R_{2\infty,0}$, $V_{1,0}$, $V_{2,0}$, γ_0; exactly how this initial problem is chosen is described in (Ge *et al.* 1994). Let A_f, B_f, ..., γ_f, denote A, B, ..., γ, in the above and define $A(\lambda)$, ..., $\gamma(\lambda)$, as

$$A(\lambda) = A_0 + \lambda(A_f - A_0), \quad \ldots, \quad \gamma(\lambda) = \gamma_0 + \lambda(\gamma_f - \gamma_0),$$

and for brevity denote them by A, ..., γ, respectively in the following. Let

$$\begin{aligned}
H_{A_c}(\theta,\lambda) &= \mathcal{P}_{12}^T\mathcal{Q}_{12} + \mathcal{P}_2\mathcal{Q}_2,\\
H_{B_c}(\theta,\lambda) &= \mathcal{P}_2 B_c V_2 + (\mathcal{P}_{12}^T\mathcal{Q}_1 + \mathcal{P}_2\mathcal{Q}_{12}^T)C^T,\\
H_{C_c}(\theta,\lambda) &= R_2 C_c \mathcal{Q}_2 + B^T(\mathcal{P}_1\mathcal{Q}_{12} + \mathcal{P}_{12}\mathcal{Q}_2)\\
&\quad + \frac{1}{2}\gamma^{-2} R_{2\infty} C_c\big[(\mathcal{Q}_{12}^T\mathcal{P}_1 + \mathcal{Q}_2\mathcal{P}_{12}^T)\mathcal{Q}_{12} + (\mathcal{Q}_{12}^T\mathcal{P}_{12} + \mathcal{Q}_2\mathcal{P}_2)\mathcal{Q}_2\big],
\end{aligned}$$

where $\mathcal{Q}$ and $\mathcal{P}$ satisfy

$$\tilde{A}\mathcal{Q} + \mathcal{Q}\tilde{A}^T + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty\mathcal{Q} + \tilde{V} = 0, \tag{13}$$

$$\big(\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty\big)^T\mathcal{P} + \mathcal{P}\big(\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty\big) + \tilde{R} = 0, \tag{14}$$

$$\mathcal{Q} = \begin{pmatrix} \mathcal{Q}_1 & \mathcal{Q}_{12} \\ \mathcal{Q}_{12}^T & \mathcal{Q}_2 \end{pmatrix}, \quad \mathcal{P} = \begin{pmatrix} \mathcal{P}_1 & \mathcal{P}_{12} \\ \mathcal{P}_{12}^T & \mathcal{P}_2 \end{pmatrix},$$

$$\tilde{R}_\infty = \begin{pmatrix} R_{1\infty} & 0 \\ 0 & C_c^T R_{2\infty} C_c \end{pmatrix}, \quad R_{1\infty} = E_{1\infty}^T E_{1\infty}, \quad R_{2\infty} = E_{2\infty}^T E_{2\infty}.$$

The nonlinear equations (corresponding to $\lambda = 1$) that result from the nonconvex constrained minimization procedure (see (Ge *et al.* 1994) for details) are

$$\rho(\theta,\lambda) \equiv \begin{pmatrix} [H_{A_c}(\theta,\lambda)]_{\mathcal{I}} \\ \text{Vec}\,[H_{B_c}(\theta,\lambda)] \\ \text{Vec}\,[H_{C_c}(\theta,\lambda)]_{\mathcal{T}.}) \end{pmatrix} = 0.$$

To guarantee a full rank Jacobian matrix along the whole homotopy zero curve except possibly at the solution corresponding to $\lambda = 1$, define the homotopy map to be $\hat{\rho}(\theta,\lambda) = \lambda\rho(\theta,\lambda) + (1-\lambda)(\theta - \theta_0)$. The Jacobian matrix of $\hat{\rho}$ is given by

$$D\hat{\rho}(\theta,\lambda) = \big(\lambda D_\theta\rho(\theta,\lambda) + (1-\lambda)I, \rho(\theta,\lambda) + \lambda D_\lambda\rho(\theta,\lambda) - (\theta - \theta_0)\big).$$

$D_\theta\rho(\theta,\lambda)$ and $D_\lambda\rho(\theta,\lambda)$ are derived from the following. Define

$$\begin{aligned}
\hat{H}_{A_c}\big(\mathcal{P}^{(j)},\mathcal{Q}^{(j)}\big) &= \mathcal{P}_{12}^{T(j)}\mathcal{Q}_{12} + \mathcal{P}_{12}^T\mathcal{Q}_{12}^{(j)} + \mathcal{P}_2^{(j)}\mathcal{Q}_2 + \mathcal{P}_2\mathcal{Q}_2^{(j)},\\
\hat{H}_{B_c}\big(\mathcal{P}^{(j)},\mathcal{Q}^{(j)}\big) &= \mathcal{P}_2^{(j)}B_c V_2 + \big(\mathcal{P}_{12}^{T(j)}\mathcal{Q}_1 + \mathcal{P}_{12}^T\mathcal{Q}_1^{(j)} + \mathcal{P}_2^{(j)}\mathcal{Q}_{12}^T + \mathcal{P}_2\mathcal{Q}_{12}^{T(j)}\big)C^T,\\
\hat{H}_{C_c}\big(\mathcal{P}^{(j)},\mathcal{Q}^{(j)}\big) &= R_2 C_c \mathcal{Q}_2^{(j)} + B^T\big(\mathcal{P}_1^{(j)}\mathcal{Q}_{12} + \mathcal{P}_1\mathcal{Q}_{12}^{(j)} + \mathcal{P}_{12}^{(j)}\mathcal{Q}_2 + \mathcal{P}_{12}\mathcal{Q}_2^{(j)}\big)\\
&\quad + \frac{\gamma^{-2}}{2} R_{2\infty} C_c\Big[(\mathcal{Q}_{12}^{T(j)}\mathcal{P}_1 + \mathcal{Q}_{12}^T\mathcal{P}_1^{(j)} + \mathcal{Q}_2^{(j)}\mathcal{P}_{12}^T + \mathcal{Q}_2\mathcal{P}_{12}^{T(j)})\mathcal{Q}_{12} + (\mathcal{Q}_{12}^{T(j)}\mathcal{P}_{12} + \mathcal{Q}_{12}^T\mathcal{P}_{12}^{(j)}\\
&\quad + \mathcal{Q}_2^{(j)}\mathcal{P}_2 + \mathcal{Q}_2\mathcal{P}_2^{(j)})\mathcal{Q}_2 + (\mathcal{Q}_{12}^T\mathcal{P}_1 + \mathcal{Q}_2\mathcal{P}_{12}^T)\mathcal{Q}_{12}^{(j)} + (\mathcal{Q}_{12}^T\mathcal{P}_{12} + \mathcal{Q}_2\mathcal{P}_2)\mathcal{Q}_2^{(j)}\Big],
\end{aligned}$$

where the superscript (j) means $\partial/\partial\theta_j$. Using the above definitions, we have for $\theta_j = (A_c)_{kl}$, where $(k,l) \in \mathcal{I}$,

$$\frac{\partial H_{A_c}}{\partial (A_c)_{kl}} = \hat{H}_{A_c}\big(\mathcal{P}^{(j)}, \mathcal{Q}^{(j)}\big),$$
$$\frac{\partial H_{B_c}}{\partial (A_c)_{kl}} = \hat{H}_{B_c}\big(\mathcal{P}^{(j)}, \mathcal{Q}^{(j)}\big),$$
$$\frac{\partial H_{C_c}}{\partial (A_c)_{kl}} = \hat{H}_{C_c}\big(\mathcal{P}^{(j)}, \mathcal{Q}^{(j)}\big),$$

for $\theta_j = (B_c)_{kl}$,

$$\frac{\partial H_{A_c}}{\partial (B_c)_{kl}} = \hat{H}_{A_c}\big(\mathcal{P}^{(j)}, \mathcal{Q}^{(j)}\big),$$
$$\frac{\partial H_{B_c}}{\partial (B_c)_{kl}} = \hat{H}_{B_c}\big(\mathcal{P}^{(j)}, \mathcal{Q}^{(j)}\big) + \mathcal{P}_2 E^{(k,l)} V_2,$$
$$\frac{\partial H_{C_c}}{\partial (B_c)_{kl}} = \hat{H}_{C_c}\big(\mathcal{P}^{(j)}, \mathcal{Q}^{(j)}\big),$$

and for $\theta_j = (C_c)_{kl}$, where $k > 1$,

$$\frac{\partial H_{A_c}}{\partial (C_c)_{kl}} = \hat{H}_{A_c}\big(\mathcal{P}^{(j)}, \mathcal{Q}^{(j)}\big),$$
$$\frac{\partial H_{B_c}}{\partial (C_c)_{kl}} = \hat{H}_{B_c}\big(\mathcal{P}^{(j)}, \mathcal{Q}^{(j)}\big),$$
$$\frac{\partial H_{C_c}}{\partial (C_c)_{kl}} = \hat{H}_{C_c}\big(\mathcal{P}^{(j)}, \mathcal{Q}^{(j)}\big) + R_2 E^{(k,l)} \mathcal{Q}_2 + \frac{\gamma^{-2}}{2} R_{2\infty} E^{(k,l)} Y,$$

where

$$Y = \big(\mathcal{Q}_{12}^T \mathcal{P}_1 + \mathcal{Q}_2 \mathcal{P}_{12}^T\big)\mathcal{Q}_{12} + \big(\mathcal{Q}_{12}^T \mathcal{P}_{12} + \mathcal{Q}_2 \mathcal{P}_2\big)\mathcal{Q}_2$$

and $E^{(k,l)}$ is a matrix of the appropriate dimension whose only nonzero element is $e_{kl} = 1$. $\mathcal{P}^{(j)}$ and $\mathcal{Q}^{(j)}$ can be obtained by solving the Lyapunov equation

$$\begin{aligned} 0 &= \big(\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty\big)\mathcal{Q}^{(j)} + \mathcal{Q}^{(j)}\big(\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty\big)^T + \tilde{V}^{(j)} \\ &\quad + \tilde{A}^{(j)}\,\mathcal{Q} + \mathcal{Q}\tilde{A}^{T\,(j)} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty^{(j)}\mathcal{Q}, \\ 0 &= \big(\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty\big)^T\mathcal{P}^{(j)} + \mathcal{P}^{(j)}\big(\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty\big) + \tilde{R}^{(j)} \\ &\quad + \big(\tilde{A}^{(j)} + \gamma^{-2}\mathcal{Q}^{(j)}\tilde{R}_\infty + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty^{(j)}\big)^T\mathcal{P} + \mathcal{P}\big(\tilde{A}^{(j)} + \gamma^{-2}\mathcal{Q}^{(j)}\tilde{R}_\infty + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty^{(j)}\big). \end{aligned}$$

Similarly for λ, using a dot to denote $\partial/\partial\lambda$,

$$\frac{\partial H_{A_c}}{\partial \lambda} = \hat{H}_{A_c}\big(\dot{\mathcal{P}}, \dot{\mathcal{Q}}\big),$$
$$\frac{\partial H_{B_c}}{\partial \lambda} = \hat{H}_{B_c}\big(\dot{\mathcal{P}}, \dot{\mathcal{Q}}\big) + \mathcal{P}_2 B_c \dot{V}_2 + (\mathcal{P}_{12}^T \mathcal{Q}_1 + \mathcal{P}_2 \mathcal{Q}_{12}^T)\dot{C}^T,$$
$$\frac{\partial H_{C_c}}{\partial \lambda} = \hat{H}_{C_c}\big(\dot{\mathcal{P}}, \dot{\mathcal{Q}}\big) + \dot{R}_2 C_c \mathcal{Q}_2 + \dot{B}^T\big(\mathcal{P}_1 \mathcal{Q}_{12} + \mathcal{P}_{12}\mathcal{Q}_2\big) + \frac{1}{2}\gamma^{-2}\dot{R}_{2\infty} C_c Y - \gamma^{-3}\dot{\gamma} R_{2\infty} C_c Y,$$

where $\dot{\mathcal{P}}$ and $\dot{\mathcal{Q}}$ are obtained by solving

$$0 = (\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty)\dot{\mathcal{Q}} + \dot{\mathcal{Q}}(\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty)^T + \dot{\tilde{V}} + \dot{\tilde{A}}\,\mathcal{Q} + \mathcal{Q}\dot{\tilde{A}}^T + \gamma^{-2}\mathcal{Q}\dot{\tilde{R}}_\infty\mathcal{Q} - 2\gamma^{-3}\dot{\gamma}\mathcal{Q}\tilde{R}_\infty\mathcal{Q},$$

$$0 = (\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty)^T\dot{\mathcal{P}} + \dot{\mathcal{P}}(\tilde{A} + \gamma^{-2}\mathcal{Q}\tilde{R}_\infty) + \dot{\tilde{R}} + (\dot{\tilde{A}} + \gamma^{-2}\dot{\mathcal{Q}}\tilde{R}_\infty + \gamma^{-2}\mathcal{Q}\dot{\tilde{R}}_\infty - 2\gamma^{-3}\dot{\gamma}\mathcal{Q}\tilde{R}_\infty)^T\mathcal{P} + \mathcal{P}(\dot{\tilde{A}} + \gamma^{-2}\dot{\mathcal{Q}}\tilde{R}_\infty + \gamma^{-2}\mathcal{Q}\dot{\tilde{R}}_\infty - 2\gamma^{-3}\dot{\gamma}\mathcal{Q}\tilde{R}_\infty).$$

The homotopy zero curve tracking algorithm (which is a standard globally convergent probability-one homotopy algorithm (Watson *et al.* 1987)) is

1) Set $\lambda := 0$, $\theta := \theta_0$.
2) Calculate $\tilde{R}$, $\tilde{R}_\infty$, $\tilde{V}$, and compute $\mathcal{Q}$ and $\mathcal{P}$ according to (13) and (14).
3) Evaluate the homotopy map $\hat{\rho}(\theta, \lambda)$ and the Jacobian of the homotopy map $D\hat{\rho}(\theta, \lambda)$.
4) Predict the next point $Z^{(0)} = (\theta^{(0)}, \lambda^{(0)})$ on the homotopy zero curve using, e.g., a Hermite cubic interpolant.
5) For $k := 0, 1, 2, \cdots$ until convergence do

$$Z^{(k+1)} = Z^{(k)} - [D\hat{\rho}(Z^{(k)})]^\dagger \hat{\rho}(Z^{(k)}),$$

where $[D\hat{\rho}(Z)]^\dagger$ is the Moore-Penrose inverse of $D\hat{\rho}(Z)$. Let $(\theta_1, \lambda_1) = \lim_{k\to\infty} Z^{(k)}$.
6) If $\lambda_1 < 1$, then set $\theta := \theta_1$, $\lambda := \lambda_1$, and go to step 2).
7) If $\lambda_1 \geq 1$, compute the solution $\bar{\theta}$ at $\lambda = 1$.

4 The parallel algorithm

In the above algorithm, Step 2) involves solving one Riccati equation and one Lyapunov equation. The Riccati equation is solved using Laub's Schur method (Laub 1979). The algorithm of Bartels and Stewart (Bartels and Stewart 1972) is applied to solve the Lyapunov equation. Although both algorithms are $O((n + n_c)^3)$, the Riccati equation, being more complicated, takes much more CPU time to solve. Once $\mathcal{Q}$ and $\mathcal{P}$ are obtained, the homotopy map $\hat{\rho}$ is formed by matrix multiplication operations.

The major part of the computation in Step 3) is that of the Jacobian matrix. The number of variables including λ in this formulation is $n_c(m + l) + 1$. Each column of the Jacobian matrix corresponds to the derivative of the homotopy map with respect to one variable and requires the solution of two Lyapunov equations (Ge *et al.* 1994). Therefore the time complexity of the Jacobian matrix computation is $O(n_c(m + l)(n + n_c)^3)$. The Bartels and Stewart algorithm finds the real Schur form of $\tilde{A}$ or $\tilde{A}^T$ depending on the Lyapunov equation. At each step along the homotopy path unnecessary factorization can be avoided if the previous factorization results from the computation of $\hat{\rho}$ and $D_\lambda\hat{\rho}$ are used.

The primary goal of this study is to make use of the existing code and to achieve reasonable parallel efficiency economically. The only part of the algorithm that is parallelized is the Jacobian matrix computation in Step 3). To utilize existing computer resources such as a network of workstations, the software package PVM (Parallel Virtual Machine) is used to provide the distributed computing capabilities.

The parallel algorithm follows the master-slave paradigm. The master sends the index of the column of the Jacobian matrix to be computed to a slave. The slave computes the corresponding column of the Jacobian, sends the column back to the master, and waits for

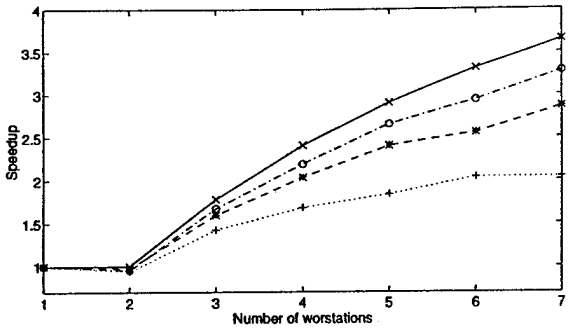

Fig. 1. Speedup with master and slaves on different machines.

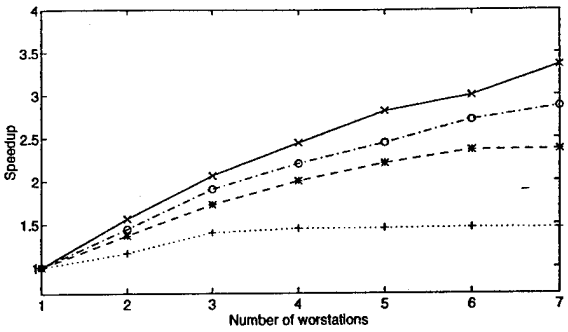

Fig. 2. Speedup when one slave is on the master machine.

the next index from the master to arrive. After receiving a column of the Jacobian, the master sends another index to the idle slave. In the implementation for the Intel Paragon, asynchronous send which sends a message without waiting for completion is used whenever possible to speed up the communication.

When the algorithm is implemented on a network of workstations, the modification to the original sequential source code consists of three parts: the first one is to spawn slave processes and set up the communication links between the master and the slaves; the second is to extract a slave program from the original code and at the same time simplify the master program; the last is to add a mechanism to guarantee correct communication between master and slaves. The first part consists of standard PVM operations, while the second is more problem oriented. To decrease communication, each slave process repeats part of the computation of $\hat{\rho}$ and $D_\lambda \hat{\rho}$ so that $\mathcal{Q}$ and $\mathcal{P}$ are not sent through the network. There is no loss of efficiency since the master is also computing the same quantities. The slave program consists of mainly the original subroutines with additional code for communication.

For the implementation on the Intel Paragon, the modification of the original code is even simpler. There is no need for a separate slave program if control statements use node identification properly. The parent process run on an Intel Paragon always gets node number 0, while other nodes are numbered 1 and higher. The statement *if node_number* == 0 precedes the code that is to be executed by the master, and an *else* following the previous master code will precede the code to be executed by the slave. The remaining modification to the original code is similar to the implementation using PVM. Asynchronous send is used whenever possible. A *wait* is used later when the data is needed, to ensure correct communication between the master and the slaves.

5 Results

The distributed code using PVM was run on a network of seven SGI Indigo^2 workstations. The data came from a control problem for suppressing vibrations in a string under transverse loading from a time varying disturbance force (Erwin 1993). For dimensions $n = 12, 20, 28, 36$ reduced order controllers of dimensions $n_c = 10, 18, 26, 34$ are sought respectively.

The speedups versus the number of workstations are shown in Figs. 1 and 2 ($n = 36$, 28, 20, 12, top to bottom). Fig. 1 shows the speedup versus the number of workstations when the master process and the slave processes are run on different machines, while Fig. 2 corresponds to the situation where the master process and a slave process with lower priority are run on one machine and the rest of the slaves are on other machines. For fair comparison

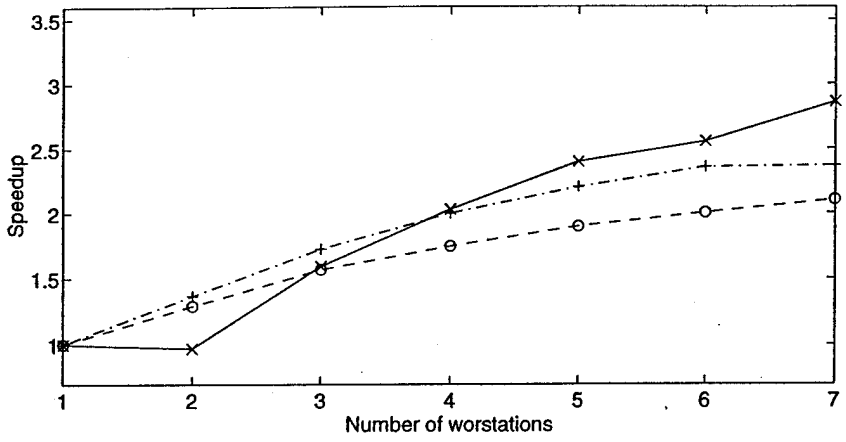

Fig. 3. Comparison of speedups.

all the speedups are computed relative to the results of the best optimized sequential code. In Fig. 1 two workstations correspond to the master process on one machine and the only slave on the other.

As shown in the figures, the speedup increases as the dimension of the problem increases, or as the number of workstations increases for a sufficiently large problem. The speedups from three scenarios (solid line—master and slaves on different machines; dash-dot line—master and a low priority slave on one machine and the rest of the slaves on others; dashed line—master and a slave with the same priority as the master on one machine) for $n = 20$ are plotted against the number of workstations in Fig. 3. If the number of workstations is < 4 it is better to use the second scenario. When the number of workstations is > 4, the speedup is higher if all the processes including the master and the slaves are run on different machines. Similar results obtain for large n.

The same algorithm is implemented using the system function calls of an Intel Paragon and run on one with 28 processors at Virginia Polytechnic Institute and State University. Fig. 4 shows the results obtained from the run for $n = 12$, 20, 28. The number of nodes varies up to 25. The highest curve corresponds to the speedup when $n = 28$ and the lowest corresponds to that when $n = 12$. As n increases, the advantage of parallel processing also increases. The highest speedup achieved for $n = 28$ using seven SGI Indigo^2 workstations is about 3.3 while the highest speedup using 25 nodes on an Intel Paragon is about 5.1. Comparing speedups is meaningful since the performance of a single SGI Indigo^2 processor is roughly comparable to that of a single i860XP Paragon node (actually, depending on the task, the 100MHz R4000 Indigo^2 is faster by a factor of 2). However, the cost of the Intel paragon is a factor of three times the cost of the SGI UNIX workstation network. Much higher speedups are potentially possible with the Paragon, but not without considerable programming effort for this controller design problem.

The above methodology can be easily generalized to industrial design environments where software packages like MATLAB or Mathematica are often used. The sequential program for mixed-norm H^2/H^∞ LQG controller synthesis has been developed into a MATLAB package. It is easy to include this distributed implementation into the MATLAB package. Installation requires two steps: the first one is to install PVM on the network of workstations, and the second is to create a file in which all the worker machines on

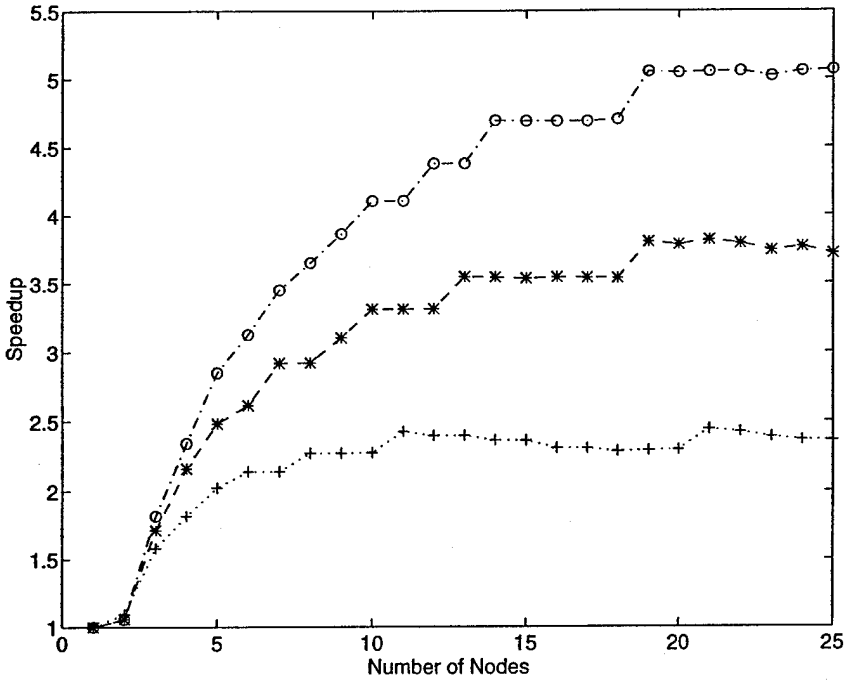

Fig. 4. Results for Intel Paragon XPE-28.

the network are listed (Geist *et al.* 1993). The execution of the distributed program from within an interactive design environment, e.g., MATLAB, can be done by using a MATLAB function defined in a MATLAB .m file, in which UNIX shell commands will start the PVM daemons if they have not already been started, and will then execute the distributed code.

In summary, parallelizing the Jacobian matrix computation in a homotopy algorithm reduces the execution time and is economical, especially for large problems. Acceptable speedups are obtained for a PVM implementation on a network of workstations. The approach can be applied to real industrial design environments to reduce controller design time and effectively utilize existing workstation networks. Compared to the cost of using real parallel computers and the cost of developing highly efficient parallel code, the approach of utilizing a network of workstations with only moderately efficient code is much more cost-effective for H^2/H^∞ controller synthesis.

References

BARTELS, R. H., AND STEWART, G. W. 1972. Solution of matrix equation $AX + XB = C$, *Comm. of the ACM*, **15**, 820–826.

BERNSTEIN, D. S., AND HADDAD, W. M. 1989. LQG control with an H_∞ performance bound: A Riccati equation approach, *IEEE Trans. Autom. Contr.*, **AC-34**, 293–305.

CANNON, R. H., JR., AND ROSENTHAL, D. E. 1984. Experiments in control of flexible structures with noncollocated sensors and actuators, *AIAA J. Guid. Contr. Dynam.*, **7**, 546–553.

ERWIN, R. S.. 1993. Private communication.

GE, Y., WATSON, L. T., COLLINS, E. G., AND BERNSTEIN, D. S.. 1994. Probability-one homotopy algorithms for full and reduced order H^2/H^∞ controller synthesis, submitted

to *Optimal Control Appl. Methods* (also see *Proc. 33rd IEEE Conf. Decision Control*, Lake Buena Vista, FL, 1994, 2672–2677).

GEIST, A., BEGUELIN, A., DONGARRA, J., JIANG, W., MANCHEK, R., AND SUNDERAM, V.. 1993. PVM 3 User's Guide and Reference Manual, ORNL/TM-12187, Oak Ridge National Laboratory, Oak Ridge, TN, 1993.

HADDAD, W. M. AND BERNSTEIN, D. S. 1990. Generalized Riccati equations for the full- and reduced-order mixed-norm H_2/H_∞ standard problem, *Systems and Control Lett.*, **14**, 185–197.

HADDAD, W. M., COLLINS, E. G., JR., AND MOSER, R.. 1993. Fixed structure computation of the structured singular value, *Proc. Amer. Contr. Conf.*, San Francisco, CA, 1673–1674.

HADDAD, W. M., COLLINS, E. G., JR., AND MOSER, R.. 1994a. Structured singular value controller synthesis using constant D-scales without D-K iteration, *Proc. American Control Conf.*, Baltimore, MD, 2819–2823.

HADDAD, W. M., COLLINS, E. G., JR., AND MOSER, R.. 1994b. Complex structured singular value analysis using fixed-structure dynamic D-scales, *Proc. IEEE Conf. Decision and Control*, Lake Buena Vista, FL, 3003–3008.

LAUB, A. J. 1979. A Schur method for solving algebraic Riccati equations, *IEEE Trans. Aut. Contr.*, **AC–24**, 913–921.

LY, U.-L., BRYSON, A. E., AND CANNON, R. H. 1985. Design of low-order compensators using parameter optimization, *Automatica*, **21**, 315–318.

WATSON, L. T., BILLUPS, S. C., AND MORGAN, A. P. 1987. HOMPACK: a suite of codes for globally convergent homotopy algorithms, *ACM Trans. Math. Software*, **13**, 281–310.

Appendix D

"An Object-oriented Approach to Semidefinite Programming"

An Object-oriented Approach to Semidefinite Programming

Yuzhen Ge† Layne T. Watson‡ Emmanuel G. Collins, Jr.*

Abstract

An object-oriented design and implementation of a primal-dual algorithm for solving the semidefinite programming problem is presented. The advantages of applying the object-oriented methodology to numerical computations, in particular to an interior point algorithm for semidefinite programming, or for solving other types of linear matrix inequalities are discussed. One object-oriented design of the primal-dual algorithm and its implementation using C++ is presented. The performance of the C++ implementation is compared with that of a procedural *C* implementation, and while the performance of the C++ implementation is comparable to that of the *C* implementation, the resulting code is easier to read, modify, and maintain.

Key words: linear matrix inequality, object oriented, primal dual algorithm, scientific computation, semidefinite programming

CR categories: D.2.2, G.1.6

1 Introduction

Object-oriented design and programming has been a major theme in software engineering in recent years. Traditional design or classical design, which has been the main software design paradigm until about mid 80's, concentrates on the actions that a system has to take and decomposes the system into separate units or modules according to their functionalities. In object-oriented design a system to be modeled is viewed as a collection of objects, each of which has its own attributes and the operations performed on an object or functions acting on an object are also defined in one syntactic unit. Objects communicate by passing messages or by calling functions from other objects which provide services. Object-oriented design is developing an object-oriented model of a system and can be realized (implemented) by object-oriented programming using languages such as C++, FORTRAN 90, or Smalltalk.

The advantages of object-oriented design and programming have been described widely elsewhere [1]. A short summary will be provided here. First, an object is an independent entity that is encapsulated in one syntactic unit. The definition of an object consists of the definition of the attributes of the object along with operations that can be performed on the object and the services or function calls provided by the object. Encapsulation hides the implementation details of an object and makes the program easier to read and modify.

The work of Yuzhen Ge and Emmanuel G. Collins, Jr. was supported in part by Air Force Office of Scientific Research grant F49620-95-1-0244 and the work of Layne T. Watson was supported in part by Air Force Office of Scientific Research grant F49620-92-J-0236 and Department of Energy grant DE-FG05-88ER25068/A004.

† Department of Mathematics and Computer Science, Butler University, Indianapolis, IN 46208.

‡ Departments of Computer Science and Mathematics, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061-0106.

* Department of Mechanical Engineering, Florida A&M-Florida State University, Tallahassee, FL 32310-

Any subsequent change to the program can be localized, making the resulting program more easily maintained.

The second advantage is information hiding. Definitions of an object which need not be known to other objects are inaccessible to other objects, preventing them from being changed accidentally. In other words, information hiding makes implementation details of an object inaccessible to other objects. However, the designer has the freedom to decide what to hide and what not to hide.

The third advantage is code reuse. Inheritance enables the definition of a new object, which can be viewed as a subclass of an existing object, without having to repeat some of the details. The new object can inherit attributes or operations from its ancestor. Inheritance is one way to support reuse of existing objects. There are different kinds of reuse in object-oriented programming; inheritance is only one of them.

One of the most popular object-oriented programming languages is C++ [11], which is used to implement the algorithm of this paper. Some of the reasons why C++ is so widely used are upward compatibility with C, design emphasis on efficiency and performance, and the availability of many useful libraries and tools. For instance, the Gnu C++ compiler and other tools are available on a wide range of platforms and provide good performance, programming environments, and reasonable compliance with ANSI standards.

There are many available libraries such as IML++[6], SparseLib++ [5] [9], STL[10] [8], and others which emphasize numerical computation. One notable package is LAPACK++, developed by Dongarra et al. [4], which is a C++ interface to LAPACK and BLAS. Ref. [4] has shown that performance of programs using the package is comparable to calling LAPACK and BLAS directly, and can at the same time reap the benefits of object-oriented programming.

This paper contains the result of object-oriented design and implementation of an algorithm for semidefinite programming. Semidefinite programming refers to minimizing a linear function subject to a linear matrix inequality [12]. That is,

$$\begin{aligned} &\underset{x \in \mathbf{R}^m}{\text{minimize}} \ c^T x \\ &\text{subject to } F(x) \geq 0, \end{aligned} \tag{1}$$

where

$$F(x) \equiv F_0 + \sum_{i=1}^{m} x_i F_i,$$

$c \in \mathbf{R}^m$, and $F_0, \ldots, F_m \in \mathbf{R}^{n \times n}$ are symmetric matrices. $F(x) \geq 0$ means that $F(x)$ is positive semidefinite.

Many problems in controls engineering can be cast in terms of a semidefinite programming problem [12]. Since a semidefinite programming problem is a convex optimization problem, which can be solved by interior point methods [7], it has attracted the attention of many researchers in interior point methods. There is a C implementation of a primal-dual algorithm for solving the semidefinite programming problem [13]. A C++ implementation of that primal-dual algorithm for the semidefinite programming problem is developed here to explore the possible benefits of object-oriented design. Because of the similarity of the primal-dual algorithm with other interior point algorithms for solving the semidefinite programming problem, the design and implementation methodology developed here can be easily modified and applied to other interior point algorithms.

The performance of a C++ implementation of the primal-dual algorithm for semidefinite programming is compared with the existing C implementation of the same algorithm from [13]. While the CPU times of the two implementations are comparable to each other, the C++ version offers the advantages mentioned earlier in this section. Segments of the code will be used to illustrate object-oriented features of the implementation.

Section 2 briefly sketches the primal-dual algorithm for semidefinite programming that will be used to illustrate the object-oriented design methodology. In Section 3, the details of an object-oriented design of the primal-dual algorithm will be given. The implementation using C++ will be described in Section 4. Comparison and discussion of the C and C++ results will be given in Section 5.

2 Primal-dual algorithm for semidefinite programming

The primal-dual algorithm for solving the semidefinite programming program given in detail in [12] will be described briefly here. The dual problem associated with the semidefinite program (1) is

$$\begin{aligned} &\underset{Z\in\mathbf{R}^{n\times n}}{\text{maximize}} \; -\text{tr}\, F_0 Z \\ &\text{subject to tr}\, F_i Z = c_i, \quad i = 1,\ldots,m \\ &\qquad Z = Z^T, Z \geq 0. \end{aligned} \tag{2}$$

The primal-dual method can be interpreted as solving the primal-dual optimization problem

$$\begin{aligned} &\underset{x\in\mathbf{R}^m, Z\in\mathbf{R}^{n\times n}}{\text{minimize}} \; c^T x + \text{tr}\, F_0 Z \\ &\text{subject to tr}\, F_i Z = c_i, \quad i = 1,\ldots,m \\ &\qquad F(x) \geq 0, Z \geq 0, Z = Z^T, \end{aligned} \tag{3}$$

where the objective function $c^T x + \text{tr}\, F_0 Z \equiv \eta$ is called the duality gap, which has the known optimum value of zero for a convex problem. The advantage of using the primal-dual formulation is that at each step information from the dual problem can be used to obtain a good update for the primal variables.

One of the methods to solve the primal-dual optimization problem is the potential reduction method. Define a potential function

$$\phi(x, Z) \equiv (n + \nu\sqrt{n}) \log(\text{ tr } F(x)Z) - \log\det F(x) - \log \text{ det } Z - n \log n,$$

where $\nu \geq 1$ is a weighting parameter in the potential. The duality gap is

$$\eta \leq \exp\left(\frac{\phi}{\nu\sqrt{n}}\right),$$

which is small if the potential function ϕ is small and approaches 0 if ϕ is approaching $-\infty$.

The whole algorithmic process can be described as follows. Starting from a strictly feasible x_0 and Z_0, find x_k and Z_k so that the potential ϕ is reduced at each step by at least a fixed amount δ in every step

$$\phi(x^{(k+1)}, Z^{(k+1)}) \leq \phi(x^{(k)}, Z^{(k)}) - \delta$$

until the duality gap η is smaller than some specified $\epsilon > 0$. The first x_k and Z_k which make $\eta < \epsilon$ is the approximate numerical solution of the primal-dual problem.

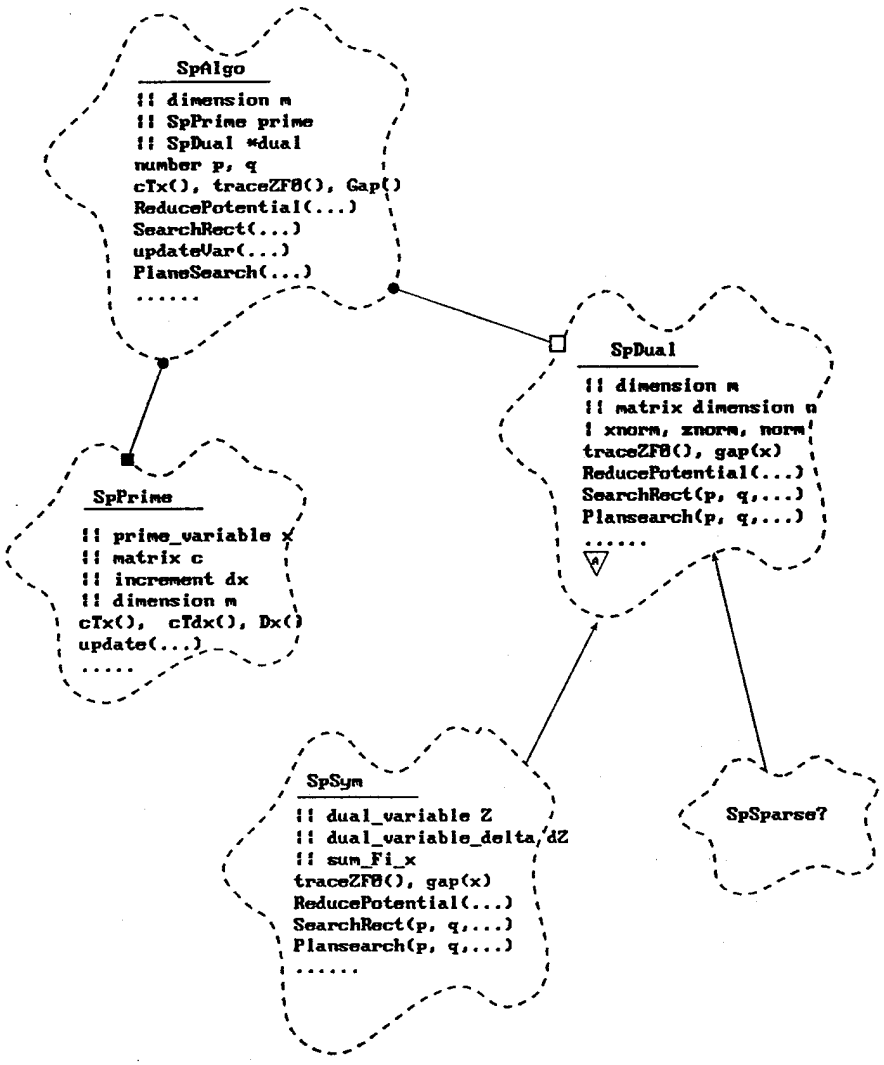

Fig. 1. Class diagram.

The primal-dual potential reduction method for solving the semidefinite program (1) can be summarized as follows. Given strictly feasible x and Z, while duality gap $\eta = c^T x + \text{tr } F_0 Z > \epsilon$ do

1. compute a direction δx for the primal variable x and a direction δZ for the dual variable Z;
2. find p, q that minimize $\phi(x + p\delta x, Z + q\delta Z)$, where (p, q) are constrained to some search rectangle in the plane;
3. update $x := x + p\delta x$ and $Z := Z + q\delta Z$.

Details of each of these steps is given in [12].

3 The object-oriented design

In C++ terminology, the word "class" is used to mean a type of object. For example, a class *Symm* can be defined for symmetric matrices, while a specific symmetric matrix is an

object of type *Symm* or an instance of the class *Symm*. C++ terminology will be followed in the rest of the paper.

Object-oriented analysis and design is one of the most active research areas for both academics and industrial practitioners. When applied in different circumstances, different analysis and design techniques may be emphasized. One of the most influential analysis and design techniques is due to Booch[1], whose conventions will be loosely followed in this paper.

First we will describe the classes used in the design and their relationship. The class diagram for the problem is shown in Fig. 1, where all the classes are defined with their attributes and functions. For clarity, only main attributes and functions are shown and the functions names may be different from that used in real implementation.

In Fig. 1, each dotted cloud-shaped figure describes a class, with the private members of the class preceded by || and the protected members of the class preceded by |. *Protected* properties are inherited by and accessible to subclasses, whereas *private* properties are not. All other properties are considered public, i.e., accessible by other classes. Protected and private properties are not accessible to other classes. For example, in the class *SpPrime*, the private attributes are *prime_variable* x, *matrix* c, and *increment* dx, the main public functions are *cTx()* which computes $c^T x$, *cTdx()* which computes $c^T dx$, *Dx()* which computes dx, and *update()* to update x.

The connection •——■ from *SpAlgo* to *SpPrime* denotes the physical containment of *SpPrime* in *SpAlgo*, while the connection •——□ from *SpAlgo* to *SpDual* denotes a pointer reference to the class *SpDual* by *SpAlgo*, which can be illustrated by the corresponding part of the definition of *SpAlgo*

```
.....
  private:
    SpPrime prime;   // primal space
    SpDual  *dual;   // dual space, dynamically allocated
    .....
```

This code declares *prime* as an instance of the class *SpPrime*, and *dual* as an instance of the class *SpDual*.

An arrow → from *SpSym* to *SpDual* denotes that *SpSym* is a subclass of *SpDual* and inherits the public and protected attributes and functions from *SpDual*. An upside-down triangle with *A* in *SpDual* denotes that *SpDual* is an *abstract class*. Many functions in *SpDual* are defined as *virtual* so that dynamic binding is used for these functions, i.e. the decision on which implementation to use is made at run time, it it cannot be determined at compile time. Consequently, if a class other than *SpSym* is used to implement *SpDual*, then the run time system will choose the right function depending on the parameters passed.

The symmetry between the primal and dual spaces in the objective function in (3) suggests that these two spaces should be fundamental classes in the problem, and so two classes, *SpPrime* and *SpDual*, are defined. *SpPrime* is relatively simple. The primal variable x is the independent variable. The only important actions on the class are to calculate $c^T x$ and update x. The class *SpDual* will contain more functions and is more complicated. It will not only calculate $\mathrm{tr} F_0 Z$, but also compute δZ, p, q, and update Z. The dual variable Z is a symmetric matrix. In the future we may want to exploit any special structure the matrix Z may have. For example, we could exploit the sparsity of the matrix Z by defining and using another class *SpSparse*. So *SpDual* is designed as an "envelope" class [3] to provide a generic dual space interface and to isolate it from the "letter" class that implements the dual space for a particular representation. As of this writing, we implemented *SpSym*, a

symmetric matrix representation not exploiting any other special structure that the matrix may have.

SpAlgo is the base class for algorithm implementation. It mediates the communication between *SpPrime* and *SpDual*, executes the computational tasks, and changes the "states" of the *SpPrime* and *SpDual* objects.

Finally *main* is a utility class, a "free" program not tied to any particular class, that stores the convergence criterion and is the driver for the algorithm. It is denoted with a dotted cloud with a shadow in Fig. 2 and Fig. 3.

The major advantage of this way of thinking is that the implementation of the dual space is completely independent of the program structure and state transition. This is desirable because the dual space implementation is where algorithmic changes are likely to occur, and this separation of the interface and implementation localizes changes in the program. The envelope class *SpDual* provides a clean and effective interface for the dual space; *SpSym* is one of the possible subclasses of *SpDual* that does nothing but implement the specification of *SpDual*. The matrix and vector classes are from LAPACK++, and are not shown in the diagram.

There are two execution scenarios, whose outlines are shown in Figs. 2 and 3, corresponding to different stages of the primal-dual algorithm.

A small square with F inside on the border of *SpDual* and *SpPrime* on the line from *SpAlgo* denotes that *SpPrime* and *SpDual* are part of the client object, *SpAlgo*. The two square boxes with L inside on the border of *SpAlgo* on the line from *main* denote that *SpAlgo* is a locally declared object in *main*. An arrow with an empty circle indicates that results or services are returned to the object pointed to by the arrow. An arrow denotes the direction of a message from one object to another. The number in front of the message denotes the execution sequence of the message.

In the first scenario (Fig. 2):

Step 1. If a search rectangle exists, *main* sends a message to *SpAlgo* to do a plane search, i.e., find p and q such that $\phi(x+p\delta x, Z+q\delta Z)$ is minimized, where p and q are constrained in the search rectangle. Otherwise *main* sends a message to *SpAlgo* to calculate the duality gap.

Step 2. *SpAlgo* sends a message to *SpPrime* to get the primal variable.

Step 3. *SpAlgo* sends the primal variable to *SpDual* along with a message to calculate the duality gap.

Step 4. *main* checks the convergence criterion. If the criterion is met, execution stops. Otherwise, go to the second scenario.

In the second scenario (Fig. 3):

Step 1. *main* sends a message to *SpAlgo* to do a potential reduction calculation.

Step 2. *SpAlgo* sends a message to *SpDual*, which returns dx, the increment for the primal variable.

Step 3. *SpAlgo* sends dx to *SpPrime* to store it in *SpPrime*.

Step 4. *main* sends a message to *SpAlgo* to calculate the search rectangle.

Step 5. *SpAlgo* sends a message to *SpPrime* to calculate $c^T dx$.

Step 6. *SpAlgo* sends $c^T dx$ along with message *SearchRect*(...) to *SpDual* to calculate and return several intermediate variables that mark the search rectangle.

Step 7. *main* checks the convergence criterion using the corners of the search rectangle. If the convergence criterion is met, execution stops. Otherwise, go to the first scenario.

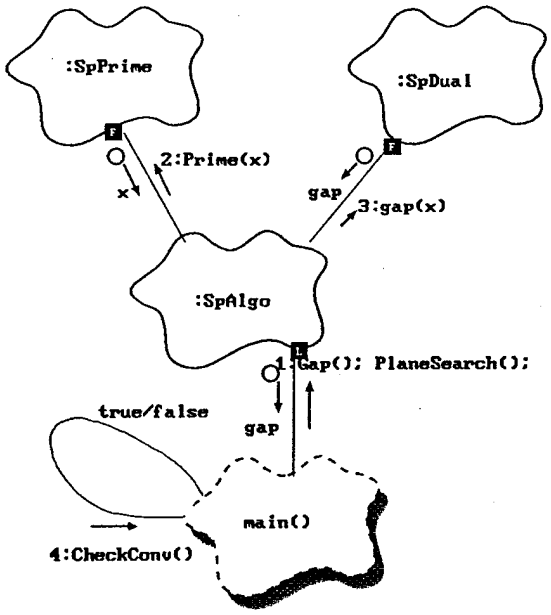

Fig. 2. Execution Scenario 1.

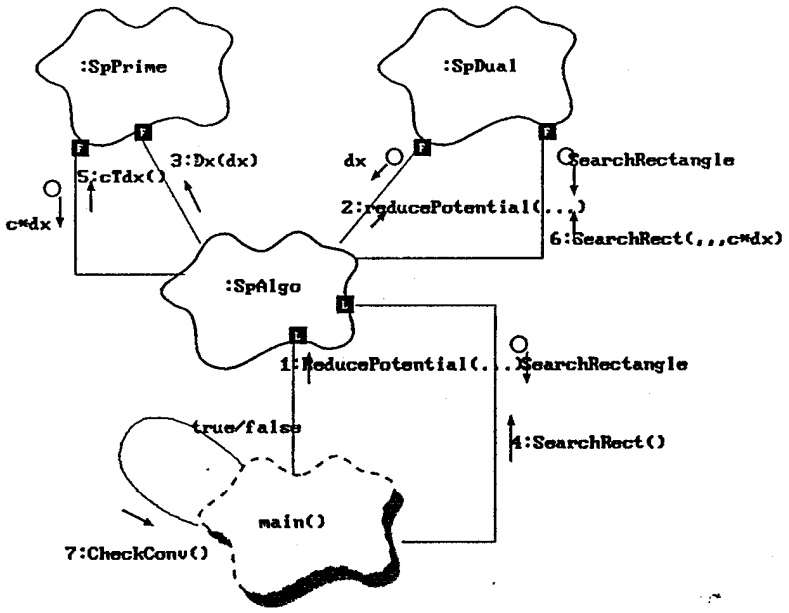

Fig. 3. Execution Scenario 2.

4 The C++ implementation

The program is built upon the LAPACK++ v1.0 package, especially the LaVectorDouble, LaGenMat, LaSymmMat classes, and BLAS++ is used extensively. Several special purpose routines are added to the LAPACK++ package to accommodate the primal-dual

```
for (k=0, pos4=pos;  k<blck_szs[i];  pos4+=blck_szs[i]-k, k++)
 {
      scal = sigx[k];
      rhs[pos4] = (1.0/scal + rho*scal)/sqrt2;
 }
for (j=0, pos=0;  j<m;  j++)
   for (i=0, pos2=0;  i<L;
         pos += blck_szs[i]*(blck_szs[i]+1)/2,
         pos2 += blck_szs[i]*blck_szs[i], i++)
    {
        /* compute V' * Fj(i) * V, store in Fsc+pos, V is scaled.*/
        cngrncb(2, blck_szs[i], F+sz+pos, R+pos2, Fsc+pos, temp);
        /* correct diagonal elements */
        for (k=0, pos4=pos;  k<blck_szs[i];  pos4 += blck_szs[i]-k, k++)
        Fsc[pos4] /= sqrt2;
     }
/*
 * solve least-squares problem; need workspace of size m + nb*sz
 * - rhs is overwritten by dx
 * - in first iteration, estimate condition number of Fsc
 */
dgels_("N", &sz, &m, &int1, Fsc, &sz, rhs, &sz, temp, &ltemp,
       &info2);
```

Fig. 4. A segment of C code.

```
for ( int i = 0, pos=0; i < j; pos += j-i, i++)
{
   double scal = sigx(i);
   dx(pos)=  ( 1.0/scal +rho*scal) * sq2;
}
/* loop over Fi,  compute V' * Fi * V, store it in Fsc */
for ( vector < LaVectorDouble>::iterator i = Fi->begin()+1;
     i < Fi->end(); i++, n++)
{
      LaVectorDouble tmp(Fsc.addr()+pd_sz*n, 1, pd_sz);
      DualScale(0, *i, vecx, tmp);
      /* correct diagonal elements */
      for ( int k = 0, pos=0; k < j; pos += j-k, k++)
        tmp(pos) *= sq2;
}
LaLeastSquare(dx, Fsc, &n);
```

Fig. 5. A segment of C++ code

algorithm for semidefinite programming. The program also uses the iterator object in STL (Standard Template Library)[10] [8] to traverse arrays of objects.

The first major difference between the C and C++ implementations is the way the initial data is read in. Unlike C/Fortran style subroutines, in which one can pass a pointer/address for a piece of storage and let the subroutine split the storage into pieces to get the data, C++ objects' constructors have no such scheme. Initialization is done by reading a data file.

The second major difference is that because C++'s objects are higher level abstractions, the implementation in C++ is less dependent upon pointer arithmetic, as shown by the code segments in *C* and C++ (Figs. 4 and 5) for doing the same computation. There is overhead associated with this higher level of abstraction, but we will show that the effect on performance is negligible.

5 Comparison and discussion of results

Two sets of data are obtained by randomly generating all the matrices $F_0, F_1, \cdots, F_m$, and the vector c. Strictly feasible initial points x_0 and Z_0 are also generated. The timing results are shown in Table 1. All the timings are done on a HP 712/60 workstation. Both the *C* implementation from [13] and the C++ implementation are compiled using the Gnu *C*/C++ compiler version 2.7.2 with the same compiler options. It is clear from Table 1 that the performance penalty for using C++ is only a few percent and decreases as the problem size increases.

TABLE 1. COMPARISON OF IMPLEMENTATIONS.

	Example 1, $n = 40$	
	C++ implementation	*C* implementation
m	time (sec)	time (sec)
20	4.7	4.4
30	6.9	6.5
	Example 2, $n = 100$	
50	229	222
75	390	381

We have shown an objected-oriented design and implementation of a semidefinite programming algorithm. Even though object-oriented technology is being used more and more widely in industry now, there are not many realistic applications to numerical computation. The programming environments and tools seem to be mature enough to apply this new methodology, and the performance seems to be comparable to a non-object-oriented implementation.

However, there are other considerations that have to be taken into account when applying object-oriented technology. First, it takes time and effort to learn the new methodology. Second, it is not a trivial task to set up the environment: compiling all the C++ packages, and verifying that they work correctly, especially when most of the C++ packages for numerical computation are still in the testing stage. Third, the resulting code size, i.e., the size of the C++ executable, is about 2.5 times that of the *C* executable. With continuing development of object-oriented technology and of compilers for object-oriented languages, the second problem, will likely be alleviated. The hardware considerations of the third problem are becoming less of a hindrance with the advances in the computer industry. We do believe that the benefits of using object-oriented methodology outweight the currently existing disadvantages.

The design and analysis in this paper can be generalized to apply to object-oriented design and implementation of other interior point algorithms which use potential reduction to find the optimum.

References

[1] G. Booch, *Object Oriented Design with Applications, second edition*, Benjamin/Cummings, Redwood City, CA, 1994.

[2] S. Boyd, L. Vandenberghe, and M. Grant, *Efficient Convex Optimization for Engineering Design*, Proc. of the IFAC Symposium on Robust Control Design, Rio de Janeiro, Brazil, Sept. 1994, pp. 14–23.

[3] J. O. Coplien, *Advanced C++, Programming Styles and Idioms*, Addison-Wesley, Redwood City, CA, 1992.

[4] J. Dongarra, R. Pozo, and D. Walker, *LAPACK++: A Design Overview of Object-Oriented Extensions for High Performance Linear Algebra*, Proc. Supercomputing '93, IEEE Press, 1993, pp. 162–171.

[5] J. Dongarra, A. Lumsdaine, R. Pozo, K. Remington, *A Sparse Matrix Library in C++ for High Performance Architectures*, Proc. Second Annual Object-Oriented Numerics Conference, 1994, pp. 214–218.

[6] J. Dongarra, A. Lumsdaine, R. Pozo, and K. A. Remington, *IML++ Iterative Methods Library Reference Guide*, http://gams.nist.gov/acmd/Staff/RPozo/sparselib++.html, 1994.

[7] Y. Nesterov and A. Nemirovsky, *Interior-point Polynomial Methods in Convex Programming*, Studies in Applied Mathematics, vol. 13, SIAM, Philadelphia, PA, 1994.

[8] D. R. Musser and A. Saini, *C++ Programming with the Standard Template Library*, Addison-Wesley Professional Computing Series, Reading, MA, 1996.

[9] R. Pozo, K. A. Remington, and A. Lumsdaine, *SparseLib++ v. 1.3, Reference Guide*, http://gams.nist.gov/acmd/Staff/RPozo/sparselib++.html, 1994.

[10] A. Stepanov and M. Lee, *The Standard Template Library*, http://www.cs.rpi.edu/projects /STL/stl-new/stl-new.html, 1993.

[11] B. Stroustrup, *The C++ Programming Language, 2nd ed.*, Addison-Wesley, Reading, MA, 1991.

[12] L. Vandenberghe and S. Boyd, *Semidefinite Programming*, SIAM Review, vol. 38, 1996, pp. 49–95.

[13] L. Vandenberghe and S. Boyd, *Software for Semidefinite Programming, User's Guide*, preprint, 1996.